VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MOBILNÍ APLIKACE PRO ANONYMNÍ KOMUNIKACI
MOBILE APPLICATION FOR ANONYMOUS COMMUNICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Štefan Krajanec

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Petr Dzurenda, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Štefan Krajanec ***ID:*** 186120
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Mobilní aplikace pro anonymní komunikaci

POKYNY PRO VYPRACOVÁNÍ:

Téma práce je zaměřeno na vývoj vlastní aplikace pro anonymní komunikaci uživatelů (instant messaging) splňující požadavky bezpečnosti a ochrany soukromí na platformě mobilních telefonů. Student vyhodnotí bezpečnost současných aplikací, analyzuje možná řešení a navrhne vlastní architekturu instant messaging aplikace splňující požadavky privacy-by-default a privacy-by-design. Výstupem práce bude funkční implementace vlastního bezpečného systému pro instant messaging zvyšující ochranu soukromí uživatelů.

DOPORUČENÁ LITERATURA:

RASTOGI, Nidhi; HENDLER, James. WhatsApp security and role of metadata in preserving privacy. arXiv Prepr. arXiv1701, 2017, 6817: 269-275.

MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Aplikácie okamžitých správ zaznamenali výrazný nárast najmä v poslednom desaťročí. Internetová komunikácia je totiž lacný a pohodlný spôsob komunikácie so vzdialenými ľuďmi. Tento nárast komunikácie používateľov a výmeny údajov prostredníctvom online sveta však ovplyvňuje bezpečnosť a súkromie používateľov, ako sa nedávno ukázalo aj v prípade problémov so súkromím v aplikácii WhatsApp. V tomto článku sa najprv hodnotia otázky bezpečnosti a ochrany súkromia súčasných mobilných aplikácií na zasielanie správ. Za druhé, navrhujeme naše základné open source riešenie so zameraním na bezpečnosť, súkromie a užívateľom dávame väčšiu moc nad svojimi šifračnými kľúčmi a dátami v systéme. V prvom rade kladieme dôraz na užitočnosť a uplatnenie systému.

KĽÚČOVÉ SLOVÁ

Súkromie, Bezpečnoť, Chatová aplikácia, Android, Metadáta

ABSTRACT

Instant messaging applications noted significant grow especially in the last decade. In fact, the Internet communication is cheap and convenient way how to communicate with distant people. However, this grow of user communication and data exchange through the online world impacts user security and privacy, as it was also shown recently by WhatsApp privacy issues. Firstly this article evaluates security and privacy issues of current mobile messaging applications. Secondly, we design our basic open source solution with the focus on security, privacy, and user centric features. Furthermore, we provide proof-of-concept implementation of our system.

KEYWORDS

Privacy, Security, Instant Messaging, Android, Metadata

KRAJANEC, Štefan. *Privacy-preserving instant messaging application* . Brno, Rok, 73 s. EEICT 2021. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Petr Dzurenda, Ph.D.

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Privacy-preserving instant messaging application “ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno . . . . . .

. . . . . .

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Petrovi Dzurendovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a cenné rady.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Mobilné aplikácie sú v súčasnosti neodmysliteľnou súčasťou každodenného života ľudí po celom svete. Každý smartphone má v sebe aspoň jednu aplikáciu z kategorie IM (Instant Messaging). Táto kategória obsahuje všetky aplikácie obsahujúce chat, v ktorom je správa adresátovi doručená v reálnom čase cez internet.

V tejto práci je bližšie opísané zabezpečenie najpoužívanejších IM aplikácii spolu s rôznymi funkciami, ktorými sú špecifické. Taktiež obsahuje vysvetlenia k tomu, čo sú to metadáta, aké rôzne údaje pod tento pojem patria a ako sa dajú zneužiť. Časť práce sa venuje analýze bezpečnostných rizík na strane servera, aj na strane užívateľa.

Najpoužívanejšou aplikáciou s 1,6 miliardy aktívnymi používateľmi je aplikácia WhatsApp, čo je približne 22% svetovej populácie [1]. Súčasné aplikácie sa zameriavajú na šifrovanie samotnej správy. Na zabezpeženie šifrovanej správy sa používa väčšinou end to end šifrovanie. Na druhej strane, o užívateľovi je možné si vytvoriť profil aj bez prístupu k obsahu správ a to práve z metadát 1.1. V sieti sa často nachádza centrálny prvok - server, ktorý môže mať prístup na sledovanie komunikácie.

Cieľom práce je návrh a realizácia systému, ktorý bude umožňovať bezpečne zašifrovanú komunikáciu medzi odosielateľom a prijímateľom, bez možnosti dešifrovať správy útočníkom. Výsledný návrh by mal chrániť dáta užívateľa ako aj užívateľovú identitu a to nielen pred útočníkom, ale aj pred serverom samotným. Databáza všetkých správ odoslaných v systéme bude môcť byť verejná, lebo všetky údaje budú šifrované, takým spôsobom, že kryptogram dokáže dešifrovať len právoplatný adresát správy. Systém je navrhovaný s velkým dôrazom na užívateľskú centrickosť a mal by podporovať automatické zmazanie správ zo serveru na základe vopred určených nastavení užívateľom a to na trvalo. Logika bude umožňovať zrušenie autorizácie pre stratené zariadenie čo spôsobí, že útočník, ktorý odcudzil zariadenie, bude vidieť len kryptogramy všetkých správ a to z dôvodu, že dané zariadenie nemá právo k získaniu správneho kľúča.

V skratke navrhnutý systém by mal dbať na užívatľskú centrickosť, ochranu užívatelských dát, klonovatelnosť systému a s je tým spojený opensource.

1 Instant messaging aplikácie a ich bezpečnosť

Táto časť obsahuje detailnejšie vysvetlenie a ozrejmenie dôvodov prečo je v záujme užívateľov šifrovať svoju komunikáciu. Čo všetko je možné zistiť o užívateľovi z komunikácie, bez toho aby bolo vidieť obsah samotných správ.

1.1 Osobné údaje

Údaje, ktoré chce užívateľ o sebe utajiť sa nazývajú osobné údaje, medzi tieto údaje patria údaje: meno užívateľa, email, miesto pobytu, telefónne číslo, poloha (súradnice vďaka GPS - Global Positioning System, čas vstupu a odchodu do aplikácie, názov/IP siete, ku ktorej je zariadenie pripojené, názvy okolitých sietí, názvy okolitých bluetooth zariadení, samotný obsah. Patria tam aj informácie o tom, kto s kým komunikuje, kto a kedy konverzáciu začal a kto kedy správu prečítal. Všetky zo spomínaných informácií okrem obsahu správy, sa nazývajú metadata. Tejto problematike sa venuje táto kapitola.

Metadáta sú často odosielané buď s obsahom správy alebo pravidelne bokom. No treba brať na vedomie fakt, že hoci v metadátach nie je prenášaný samotný obsah správy, tak vystavenie metadát môže mať značný vplyv na súkromie užívateľa. Napríklad ak zamestnanec nejakej väčšiej organizácie (vláda, veľká spoločnosť) komunikuje s novinárom za účelom odhalenia nejakej pofidérnej činnosti, tak je možné na základe metadát túto komunikáciu odhaliť. Napríklad, ak sa zariadenie zamestnanca a zariadenie novinára často vyskytujú v blízkosti alebo majú v okolí rovnaké WiFi siete, prípadne bluetooth zariadenia, tak je možné predpokladať, že vlastníci zariadení spolu komunikovali aj keď si nevymieňali žiadne správy. Na základe takýchto údajov si sociálne siete môžu robiť sieť zariadení, s ktorými ste prišli do kontaktu a na jej základe navrhnúť napríklad relevantného priateľa z okolia, prípadne dokázať komunikáciu/stretnutie s danou osobou, respektíve jeho zariadením [16, 18].

1.2 Vývoj instant messaging aplikácii

V roku 1998 nebolo množstvo IM - (Instant Messaging) aplikácií, najznámejšou z nich bolo ICQ. Okrem klasických správ aplikácia neskôr obsahovala funkcie ako telefonovanie, videohovor a hranie hier. Táto aplikácia nekládla skoro žiaden dôraz na vyššie zabezpečenie užívateľov a ich správ. V top 8 najpoužívanejších aplikácií sa udržal do roku 2011IM.

Do povedomia sa začiatkom roku 2003 dostala aplikácia Skype, ktorá mala rovnaké funkcie ako ICQ no dbala trošku viac na zabezpečenie správ, okrem posielania správ umožňovala aj konferenčné hovory s možnosťou videohovoru. Skype sa drží v dolnej polovici top 8 dodnes.

Koncom roku 2009 do hry vchádza WhatsApp, aplikácia, ktorá sa snaží chrániť súkromie užívateľa. Táto aplikácia obsahuje čisto len komunikáciu cez správy, neskôr sa pridá funkcionalita hovorov. WhatsApp doteraz neobsahuje žiadne hry. Táto aplikácia je v súčasnosti najpoužívanejšou aplikáciou na IM komunikáciu.

Koncom roku 2014 vznikla instant messages applikácia od Facebooku, niesla názov Messenger. Spočiatku aplikácia ponúkala len možnosť chatovania, neskôr sa pridalo množstvo užitočných aj neužitočných funkcií. Pomerne novou užitočnou funkciou bol skupinový chat, ktorý postupne poimplementovali aj ostatné aplikácie. Aplikácia Messenger je v súčastnosti druhou najpoužívanejšou IM aplikáciou.

V roku 2018 sa do TOP 8 dostala aplikácia Telegram. Je to aplikácia, ktorá sa snaží dbať na zabezpečenie komunikácie. Táto aplikácia je najpoužívanejšia v Rusku. Ruská vláda sa snaží s touto aplikáciou bojovať. Na trhu sa vyskytujú aj aplikácie, ktoré sa snažia prioritne chrániť súkromie užívateľa. Sú to aplikácie **Threema, Ricochet, The Signal**, medzi tieto aplikácie patria aj **WhatsApp** a **Telegram**. V skutočnosti ich je omnoho viac, no toto sú najpoužívanejšie z nich.

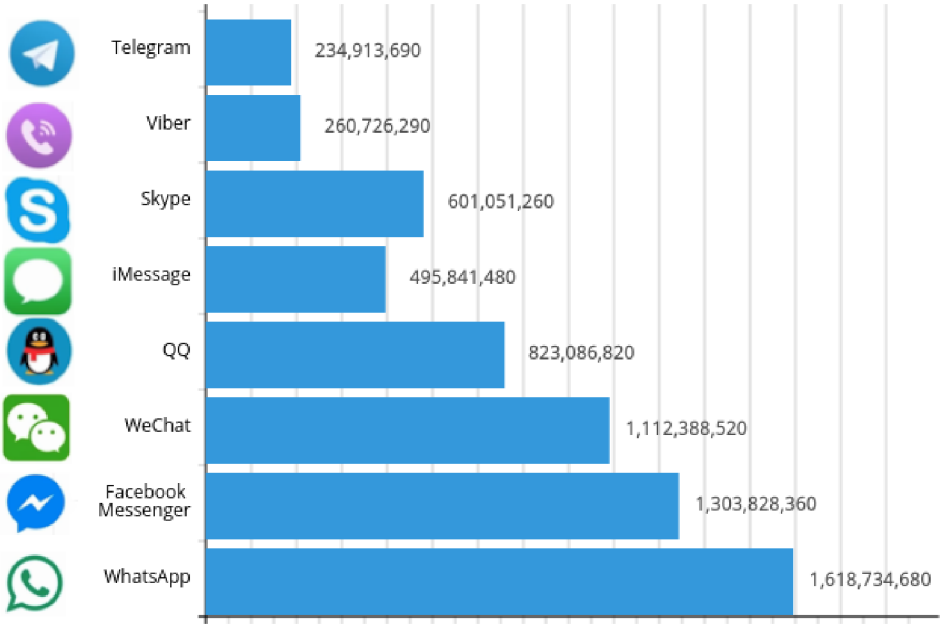
Do IM aplikácií sa radia aj aplikácie zoznamkového charakteru, ako je napríklad Badoo a Tinder. Aplikácií spadajúcich do tejto kategórie je omnoho viac, než len tie, ktoré sú spomínané v tejto práci. Z tretieho kvadrantu roku 2019 je top 8 IM aplikácií aj s počtom používateľov vyobrazených 1.1. [17].

1.3 Typy ochrany súkromia

Všeobecné nariadenie o ochrane údajov z roku 2018 prinieslo mnoho dôležitých zmien, organizácie museli spraviť mnoho, aby mohli ostať v súlade zo zákonom. Vznikli nové termíny označenia ochrany súkromia

- **Privacy by Design**

  Jedná sa typ ochrany súkromia, ktorý už od jadra návrhu uvádza, že každá akcia, ktorú spoločnosť podnikne ohľadom spracuvávania osobných údajov, sa uskutočňuje s ohľadom na ochranu súkromia v každom kroku. To zahŕňa interné projekty, vývoj softwaru, veľké IT systémy, IM aplikácie a mnoho iného. V praxi to znamená, že oddelenie, ktoré spracúva osobné údaje, musí zabezpečiť, aby bolo do systému zabudované súkromie počas celého životného systému akéhokoľvek procesu.



Obr. 1.1: Akuálny rebríček top 8 aplikácií.

- **Privacy by Default**

  Je to termín, ktorý znamená, že už v defaultnom nastavení produktu sú štandardne nastavené najprísnejšie nastavenia ochrany súkromia. Tu je dôležité slovo defaultne, totiž je množstvo nastavení a externých pluginov, ktoré zvyšujú ochranu súkromia, avšak je nutné, aby koncový užívateľ vykonal nejakú akciu aby docielil najväčšiu možnú ochranu súkromia v danej platforme. Okrem toho by sa akékoľvek osobné údaje poskytnuté užívateľom mali uchovávať LEN tak dlho, ako je potrebné na poskytnutie služby. Ak sa zverejní alebo uchová viac informácii, prípadne na dlhšie, než je nutné k požadovanému procesu, tak došlo k porušeniu "Privacy by Default".

Na základe horeuvedených definícií pojmov, boli aplikácie zaradené do tabuľky podľa príslušnosti k definíciám. V 1.1 je vidno, že máloktorá aplikácia spĺňa aspoň jedno z nariadení. Napríkladu aplikácia Messenger od Facebooku poskytuje možnosti na zlepšenie zabezpečenia súkromia, ale užívateľ si to musí zinicializovať sám a to formou použitia tajnej konverzácie. Keďže táto funkcia nie je defaultná, tak messenger právom nepatrí pod označenie Privacy by Default. Výsledná aplikácia tejto diplomovej práce by mala mať právo na zaradenie pod Privacy by Design a aj Privacy by Default, rovnako ako má Threema a WhatsApp [2].

Tab. 1.1: Zoradenie aplikácii podľa prístupu k osobným údajom

Aplikácia	Privacy by Design	Privacy by Default
WhatsApp	✓	✓
Messenger	✗	✗
Telegram	✗	✗
Threema	✓	✓
Tinder	✗	✗
Instagram	✓	✗

1.4 Užívateľská centrickosť z hladiska zabezpečenia

Jedná sa o prevzatú definíciu, ktorá šíri filozofiu prispôsobovania technologii a vzhladu aplikácii na užívateľov, aby sa im s aplikáciami príjemne pracovalo. Pre účel tejto práce sme definíciu prevzali a znamená, že užívateľ bude mať plnú moc nad správou svojich šifrovacích kľúčov pre jednotlivé zariadenia a symetrické klúče pre každú miestnosť. Užívateľ vie kedykoľvek zmazať kľúče pomocou svojho zariadenia a zmazanie kľúčov sa rozdistribuuje po systéme a dané kľúče budú zneplatnené, tým pádom všetky správy zašifrované zmazaným kľúčom budú nenávratne nedešifrovatelné. Tak ako mazanie a generovanie kľúčov, je možné aj zmeniť platnosť kľúčom. Rovnako je možné nakonfigurovať nastavenia v miestnosti podľa potrieb užívateľov v miestnosti (doba platnosti správ, neaktivita chatu, zverejnenie info oponentom, nastavenia k odosielaným médiám a podobne).

1.5 Prieskum instant messaging aplikácií

V súčasnosti je mnoho aplikácií patriacich pod toto označenie. V tejto práci budú podrobnejšie prebraté Whatsapp[5, 6, 8, 10, 7], Instagram direct, Messenger[10], Badoo[4], Tinder[11], Threema[5, 12], Telegram[5, 14], The Signal[15] a Ricochet[13]. Tieto aplikácie majú rôzne ďalšie funkcie, ktorými sa snaží udržať užívateľov.

Porovnanie základných funkcií je k nahliadnutiu v 1.2. Tabuľka obsahuje len aplikácie, ktoré sú v práci rozobrané podrobnejšie. Jedná sa o výber najpoužívanejších aplikácií vo svojej kategórii.

- **Zoznamky** - Badoo a Tinder
- **Súkromie** - WhatsApp, Threema, Richochet, The Signal a Telegram
- **Socialne siete založené na spame** - Messenger a Instagram direct

Tab. 1.2: Porovnanie funkcionalít jednotlivých IM aplikácii

	Whatsapp	Instagram Direct	Badoo	Tinder	Threema	Facebook mesenger	Ricochet	The Signal	Telegram
chat	✓	✓	✓	✓	✓	✓	✓	✓	✓
skupinový chat	✓	✓	✗	✗	✓	✓	✗	✓	✗
telefonát	✓	✓	✓	✗	✓	✓	✗	✓	✓
videohovor	✓	✓	✓	✗	✗	✓	✗	✓	✗
integrácia kontaktov z FB	✗	✓	✗	✗	✗	✓	✗	✗	✗
uprava vzhľadu chatu	✓	✗	✗	✗	✗	✓	✓	✓	✓
zasielanie súborov	✓	✗	✗	✗	✓	✓	✗	✓	✓
fotografie ku kontaktom	✓	✓	✓	✓	✓	✓	✗	✓	✓
registrácia bez telefonneho čísla	✗	✓	✓	✗	✗	✓	✓	✗	✓
registrácia cez FB	✗	✓	✓	✗	✗	✓	✓	✗	✓

Ďaľej sa bolo treba sústrediť na pozadie aplikácií, dosť to uľahčil web securemessagingapps.com, ktorý obsahuje množstvo zaujímavých informácií o nami sledovaných aplikáciách. Na základe údajov z dokumentácií a údajov zo spomínanej stránky vznikla 1.3, kde sú bližšie popísané spoločné, rozdielne a nebezpečné veci z pohľadu ochrany súkromia pre aplikácie Messenger, Threema, WhatsApp a Signal. Z tabuľky je možno vyčítať množstvo dôležitých údajov k danej aplikácií, napríklad ako je to s poskytovaním užívateľských údajov tretím stranám a priority riešiť súkromie užívateľov. S tým je previazaný aj zber a uchovávanie užívateľských dát. Nachádza sa tam aj porovnanie kryptorafických primitív použitých v danej aplikácii. Z tohto hľadiska všetci používajú bezpečné typy šifier a dĺžky kľúčov v aktuálnom období [3].

Taktiež je možné vidieť, že niektoré IM aplikácie majú bezproblémový prístup k obsahu správy, nehashujú základné osobné údaje a nešifrujú metadáta, prienikom všetkých spomínaných atribútov je Messenger od Facebooku. Táto platforma patrí, alebo najmenej dlhé roky patrila, k najrozšírenejšej IM platforme aj napriek tomu, že jej postoj k ochrane súkromia je na nízkom leveli. Ďalej spomedzi sledovaných aplikácií platformy Messenger a WhatsApp je aktívne logovanie času a IP pripojenia, čo je dosť citlivý a osobný údaj pre ochranu osobných údajov. Čo po spojení s informáciou, že platformy nešifrujú metadáta a faktom, že NIKDY nedali prístup pre nezávislú bezpečnostnú analýzu kódu platformy patriace pod Facebook nevynakladajú veľké úsilie na ochranu osobných údajov.

Riešenie problému doručenia správy od odosielateľa k adresátovi sa delí na dva typy [9]

- Synchronny prístup doručenia správ je špecifický tým, že správy sú šifrované a odosielané práve vtedy, keď je odosielateľ aj adresát online. Výhoda je v

tom, že strany si dohodnú aktuálne platný kľúč a odošlú správu priamo na zariadenie adresáta.

- Asynchronny systém je systém v ktorom nemusia byť komunikujúce strany online v čase odoslania/doručenia správy. Figuruje tu server, kde sú uložené údaje, ktoré sú potrebné pre dokončenie odoslania/prijatia a zároveň zašifrovania/dešifrovania správ.

Každý postoj má svoje výhody aj nevýhody. Synchronny systém sa javí ako bezpečnejší z dôvodu, že každá správa vie byť šifrovaná unikátnym kľúčom - dobrá forward security. Asynchronny systém umožňuje odosielanie správy aj ak je oponent offline, avšak správy šifruje rovnakým kľúčom do momentu, kedy je vygenerovaný nový kľúč, čo tento prístup robí aj rýchlejším, keďže pred každou správou nie je nutné vykomunikovať a dohodnúť nový šifrovací kľúč. Konkrétne práve spoločnosť Facebook figurovala v kauze o predaji osobných údajov miliónov užívateľov tretím stranám.

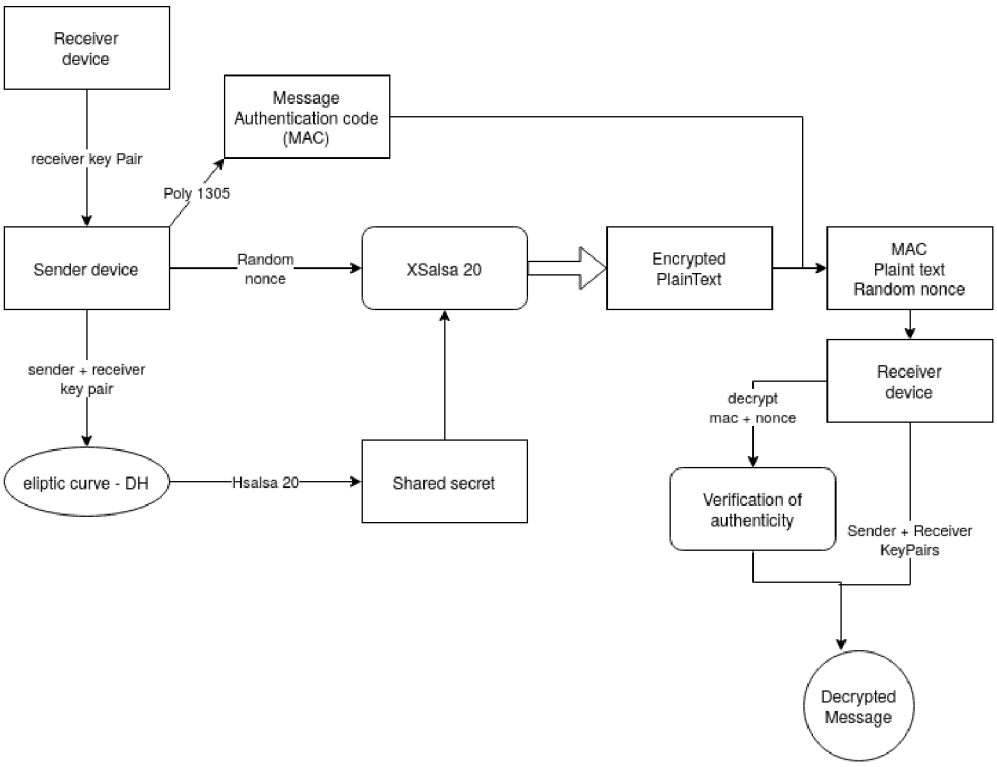
Tab. 1.3: Krypto porovnanie IM aplikácii

	Messenger	**Threema**	**WhatsApp**	**Signal**
poskytovanie dát tretím stranám	1	0	1	0
postoj k súkromiu	chabý	dobrý	chabý	dobrý
zdroje	spoločnosť	užívateia	spoločnosť	fondy
zber užívateľských dát	1	0	1	0
privacy by default	0	1	1	1
kryptografické primitíva	Curve25519/ AES-256/ HMAC-SHA256	Curve25519-256/ XSalsa20-256/ Poly1305-AES-128	Curve25519/ AES-256/ HMAC-SHA256	Curve25519/ AES-256/ HMAC-SHA256
opensource	0	0	0	1
hashovanie osobných údajov	0	1	0	1
kľúč uložený v zariadení	1	1	1	1
čitatelnosť správ pre poskytovateľa	1	0	0	0
šifrovanie metadát	0	1	0	1
sekundárny faktor overenia	0	1	1	0
šifrovanie záloh	?	1	ios - 1 android - 0	-
Logovanie času a IP	1	0	1	0
nezávislá bezpečnostná analýza kódu	nikdy	2015	nikdy	2014
samozničujúce správy	1	0	0	1

Threema

Na oficiálnych stránkach Threemy, je zverejnený cryptography whitepaper [26], kde je bližšie vysvetlené pozadie šifrovania správ a chod pozadia aplikácie. .

Na obr. 1.2 je vyobrazený diagram komunikácáe šifrovania a dešifrovania správ v platforme Threema. Prvým krokom je vygenerovanie dvojice kľúčov (SK - Súkromný kľúč a VK - Verejný kľúč) pre prijímateľa aj odosielateľa, nasledovne adresát odošle svoj VK odosielateľovi. následne z VK adresáta a SK odosielateľa sa pomocou eliptickej krivky DH - Diffie-Hellman protokol slúžiaci na dohodnutie kľúča cez verejný kanál vypočíta kľúč, ktorý je zahashovaný pomocou HSalsa20 -> Shared secret. Ďalej odosielateľ vygeneruje náhodné zoskupenie 24 bajtov (nonce) následne pomocou prúdovej šifry XSalsa20, ktorej parametre sú shared secret, nonce a text správy. Súčasne odosielateľ vytvorí message authentification code (MAC). MAC, zašifrovaný text a nonce sa odosielajú adresátovi. Adresát reverzne pomocou nonce a shared secret dešifruje MAC a pomocou svojho kľúča a klúča odosielateľa dešifruje správu.

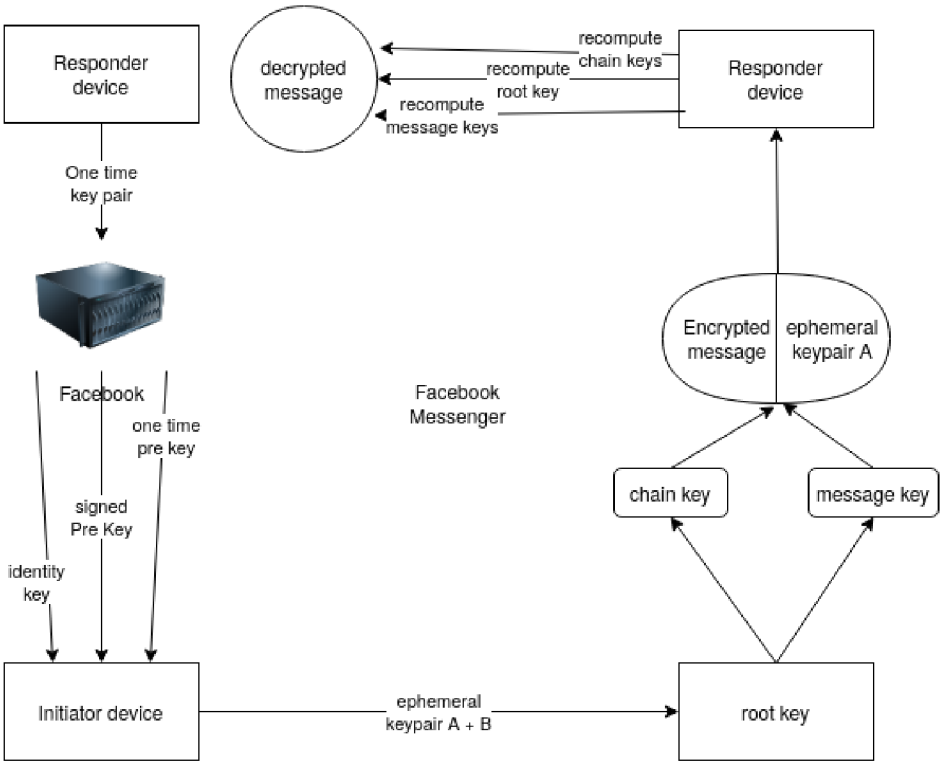


Obr. 1.2: Threema logika šifrovania a dešifrovania

Facebook Messenger - secret conversation

Na stránkach Facebooku je zverejnený whitepaper k funkčnosti secret konverzácií v aplikácii messenger od Facebooku. [27]

Na diagrame 1.3 je načrtnutá logika generovania a distribúcie kľúčov pre tajné konverzácie v messengeri od facebooku. Na začiatok adresát odošle vygenrovanú dvojicu kľúčov na server facebooku, ten pošle späť Identitiy key, Signed pre key a one time pre key, iniciátor následne vygeneruje dočasný kľúč A+B a z neho vygeneruje Root Key. Iniciátor vypočíta chain key a message key z root key. Následne sa zašifruje obsah správy pomocou chain key potom je odoslaná adresátovi spolu s message key a dočastným kľúčom A. Adresát si prepočíta chan key, root key a message key, pomocou ktorých dokáže dešifrovať správu.



Obr. 1.3: Logika šifrovania pre Facebook messenger - secret conversation

1.6 Bezpečnosť komunikácie

Svet zažil množstvo únikov osobných údajov práve z dôvodu slabého zabezpečenia komunikácie alebo zámerného dešifrovania za účelom zámerného odovzdania tretím stranám. Na ceste správy od odosielateľa k adresátovi je viacero kritických miest, ktoré je možné využiť tak, aby bolo umožnené dostať sa k obsahu správ. Niektoré z týchto miest sú do systému vložené zámerne.

1.6.1 Analýza bezpečnostných rizík

K prvému riziku prichádza už pri prihlasovaní. Ak si útočník dokáže nejakým spôsobom odchytiť autentizačný request užívateľa, v ktorom budú prihlasovacie údaje do účtu obete. Takýto request obsahuje užívateľské meno a hash z hesla. Modernou variantou je, ak sa na autentizovanom zariadení vygeneruje auth token, ktorý sa pridá do hlavičky requestu a ak je zariadenie úspešne autorizované, pridá sa daný token do databázy. Na istý čas sa tento token stáva identifikátorom prihláseného užívateľa až dovtedy kým platnosť tokenu neskončí. Ďalším levelom by mohlo byť zašifrovanie pridaného tokenu verejným kľúčom asymetrického kryptografického systému, prípadne spolu s nezašifrovaným tokenom odosielať aj autentizačný kód MAC, ktorý bude obsahovať časovú značku. Takýto request môže byť zašifrovaný verejným kľúčom z TLS - (Transport Layer Security) systému, takže server musí disponovať platným certifikátom.

1.6.2 Bezpečnosť koncových zariadení

Ak sa bezpečne podarí odoslať správu zo zariadenia na server a nik ju cestou nedokáže odchytiť a dešifrovať, tak je ďalším rizikom samotné spracovanie a ukladanie dát. Väčšina aplikácií má vlastnú databázu, kde dané správy ukladá. Bolo by veľmi nebezpečné, ak by do databázy ukladali surové dáta. Avšak aj v prípade ak by ich ukladali zašifrované je otázka, ako ich odošlú adresátovi [18].

- Symetrické šifrovanie je nutné doručiť daný kľúč prijímateľovi. Keďže správu šifruje odosielateľ a dešifruje adresát, ale obaja musia mať rovnaký kľúč, ktorý je potreba mať niekde uložený.
  - **kľúč uložený databáze** poskytovateľa znamená to, že pre správy užívateľa dešifrovateľné. K správam sa vie jednoducho dostať, je vlastníkom kľúča, ktorým sa správa šifruje a dešifruje.
  - **uloženie kľúča v zariadení prijímateľa** a odosielateľa. V tom prípade je nutné mať uložený rovnaký kľúč v oboch zariadeniach, tam je problém s distribúciou a je nutné dôverovať poskytovateľovi, že daný kľúč si on nikde neuloží. Bezpečnosť uloženia si kľúča v zariadení je opísaná ďalej.

- Asymetrické šifrovanie medzi odosielateľom a serverom, tak by mala byť rovnako šifrovaná aj smerom od serveru k prijímateľovi. Ak by nebola šifrovaná smerom od serveru k adresátovi, tak by server posielal len surové dáta, čo útočníkovi uľahčuje prácu.

  V prípade, že sa odosiela zašifrovaná správa aj od serveru k prijímateľovi, tak odosieľateľ musí poznať adresátov VK a adresát to svojím SK dokáže dešifrovať. V prípade, že odosielateľ nemá VK prijímateľa, tak používa VK serveru. Ten správu dešifruje svojím SK a následne zašifruje VK prijímateľa. To je veľký nedostatok systému. Server má k dispozícii nezašifrovanú správu. Algoritmus RSA - Rivest–Shamir–Adleman. Asymetrická šifra, používa rozdielny kľúč na šifrovanie (VK) a dešifrovanie (SK) je prvým algoritmom, ktorý je vhodný pre podpisovanie aj pre šifrovanie. Je založené na predpoklade, že nejestvuje algoritmus, ktorý si poradí s faktorizáciou veľkých čísel. je asymetrická šifra, používa rozdielny kľúč na šifrovanie (VK) a dešifrovanie (SK) je prvým algoritmom, ktorý je vhodný pre podpisovanie aj pre šifrovanie. Je založené na predpoklade, že nejestvuje algoritmus, ktorý si poradí s faktorizáciou veľkých čísel. Tento algoritmus by po vyriešení pár riešiteľných problémov umožňoval relatívne bezpečnú komunikáciu. Avšak na zašifrovanie a dešifrovanie takýchto správ treba relatívne veľký čas aj pri menších správach. Proces šifrovania a dešifrovania asymetrickou kryptografiou dokáže až moc nepríjemne predĺžiť čas od odoslania správy po zobrazenie u adresáta.

Analýza komunikácie

Zabezpečenie správ a komunikácie so serverom je individuálne pre väčšinu aplikácii. Z každej aplikácie boli odchytené komunikačné pakety pri odosielaní správy, obrázku a súboru. Odosielaná bola textová správa: VUT Brno, súbor .txt s obsahom: *VUT Brno* a obrázok, na ktorom bolo logo VUT Brno. Nie všetky aplikácie podporovali prenos týchto typov správ, čo sa v ktorej aplikácii podarilo odoslať je vyobrazené v 1.4. Meranie prebiehalo tak, že do smartfónu bola nainšalovaná aplikácia ktorá presmerovala komunikáciu cez lokálnu VPN - (Virtual Private Network) sieť, ktorej úlohou bolo logovať všetku komunikáciu rovnako ako program Wireshark. Po ukončení sledovania komunikácie, bol z telefónu presunutý súbor, ktorý bolo možné otvoriť v programe Wireshark. Vo všetkých prípadoch bola komunikácia zabezpečená TLS, jednalo sa o použitie TLS vo verzii 1.2 a 1.3. Z toho vyplýva, že nebola odhalená priamo komunikácia medzi klientom a serverom. U aplikácií Badoo a The Signal bola nájdená komunikácia podobná initializácii DH protokolu. Keďže sa nepodarilo nahliadnuť pod TLS, tak treba dôverovať údajom, ktoré sľubuje vlastník

aplikácie, takže je predpoklad, že šifrovanie typu E2E - End to End u aplikácií Whatsapp a The Signal. U aplikácie Messenger od Facebooku udávajú E2E šifrovanie len pri použití tajnej konverzácie a u aplikácie Telegram len v konverzáciach, ktoré nie sú skupinové.

Aj napriek zisteniu, že komunikácia klient-server je zabezpečená s TLS, tak stále jestvuje mnoho rizík odhalenia správy. Taktiež je dosť pravdepodobné, že server samotný môže mať prístup k obsahu správ a poskytovať ich tretím stranám prípadne spracúvať na účely cieleného marketingu či sledovania, keď nie na základe obsahu správ tak na základe metadát.

Tab. 1.4: Vyobrazenie úspešnosti odchytu paketov v jednotlivých aplikáciách

	Whatsapp	Instagram Direct	Badoo	Tinder	Threema	Facebook mesenger	The Signal
text	✓	✓	✓	✓	-	✓	✓
obrázok	✓	✓	✗	✗	-	✓	✓
súbor (.txt)	✓	✗	✓	✗	-	✗	✗

1.6.3 Riziká zabezpečenia na strane užívateľa

Predpokladajme zabezpečenú komunikáciu pomocou IM aplikácie typu Threema, Viber. Údajne tieto aplikácie neuchovávajú žiadne užívateľské správy v ich databáze. Správy by mali byť uložené len na telefónnych zariadeniach účastníkov. Samozrejme, že tieto správy sú nejakým spôsobom v telefóne zašifrované. V prípade Threemy je to užívateľom nastavené heslo a WhatsApp používa hash SHA-256 o veľkosti 256 bitov. Z čoho vznikne tento hash nie je známe, zrejme z náhodne generovaného reťazca, ktorý sa vloží do hashovacej funkcie. Tieto hashe sú buď uložené na serveri poskytovateľa a nejakým spôsobom sa dosťahuvávajú do telefónu alebo sú uložené priamo v danom zariadení. V telefónoch je pomerne jednoduché dostať sa k databáze aj a ak je akokoľvek zašifrovaná (s klúčom uloženým v zariadení), je len otázkou pomerne krátkeho času kým sa ju podarí dešifrovať. V prípade pinu alebo párznakového hesla, ktoré užívateľ zvolil je možné za krátky čas nájsť správne heslo pomocou slovníkového útoku.

Ďalším spôsobom vniknutia do niekoho správ je dostať sa k jeho telefónu a nejakým spôsobom získať kľúč na odomknutie (pin, heslo, faceId alebo odtlačok prstu) a jednoducho si prezrieť správy v aplikácii a vydávať sa za autora prípadne adresáta správ.

Jestvuje aj niekolko softwarových možností. Jednou z nich je inštalácia keylogger aplikácie do telefónu obete. Keylogger funguje tak, že sleduje a uchováva históriu

stlačených kláves. Z toho vyplýva, že je možné spätne zistiť, čo daná obeť písala vrátane neodoslaných správ. Jedná sa o jednostrannú metódu, takže ak by útočník chcel obojstrannú komunikáciu, musí podobnú aplikáciu dostať do telefónu všetkým účastníkom.

Ďalší zo spôsobov je inšpirovaný keyloggerom, nie je to doprovodná aplikácia, ktorá sleduje kliky na klávesnici, ale priamo klávesnica samotná. Ak sa podarí dostať do daných zariadení útočníkovú klávesnicu, odosielanie dát viete jednoducho skryť za štatistiky o používaní a obohacovanie globálnej slovnej zásoby pre danú klávesnicu podobne ako to má Google klávesnica.

2 Prieskum technológií

Výstupom práce by mala byť funkčná aplikácia, ktorá poskytuje čo možno najväčšiu ochranu súkromia, je preto nutné zvoliť najvhodnejšie technológie.

2.1 Technológie na serveri

Na začiatok, je potrebné vybrať technológie, ktoré budú používané na serveri. V prvom rade je treba dbať na to, že je nutné vytvoriť chatovú aplikáciu. Je nutné zvoliť technológiu podporujúcu funkciu streamov, ktorá je na rozdiel od REST viac využiteľná v IM aplikácii. Pod streamom je možné rozumieť jeden request, na ktorý môže prísť odpoveď viac než jedenkrát. Ku príkladu, po otvorení chatu sa nečítajú historické správy a zobrazia sa, po tom, čo oponent odošle novú správu, je táto správa ce stream údajov odoslaná na zariadenia a pridaná do kolekcie správ. Rozdiel je v tom, že najskôr sa stiahnu všetky správy a všetky nové automaticky prichádzajú. Keby nebola používaná technológia streamov, tak by bolo nutné pravidelne sa dotazovať na server a kontrolovať, či medzi jednotlivými dotazmi neboli prijaté nové správy. Najvyužívanejšie z podobných technológii je

- **GRPC** [19] - opensource vytvorený spoločnosťou Google, jedná sa o jazyk pre dotazy (API - (Application programming interface)). Táto technológia nie je ťažká na implementáciu v symbióze s množstvom technológií. Jedná sa o rýchlu komunikáciu, ktorá je obalovaná vlastným protokolom bez ďalšej réžie. Takže šetrí mnoho času na jednotlivých vrstvách siete, preto je tak rýchla. Medzi výhody patrí to, že je jednoducho implementovaná, množstvo kódu, ktorý je treba sa sám vygeneruje a do zariadení je ho treba len skopírovať. Nie je vhodný do projektov, kde je množstvo závislostí a časté zásahy/úpravy do už naprogramovaných vecí.
- **GraphQL** [20] - opensource vytvorený spoločnosťou Facebook, jedná sa o jazyk pre dotazy - API. Technológia je výbornou voľbou pre väčšie projekty, kde je nutné sťahovať viackrát rovnaké dáta. Výborný spôsob pre optimalizáciu requestov, keďže to, čo klient od servera chce si sám klient požiada, nemusí sťahovať zbytočne redundantné dáta a tým zaťažovať server a plytvať zdrojmi. Implementácia tejto technologie do projektu je spočiatku ťažšia, lebo je treba napísať množstvo kódu aj na strane servera aj klienta (u GRPC sa len vygeneruje). Ak sa jedná o projekt, ktorý bude neskôr platforma alebo bude do neho pridávané veľké množstvo nových závislých a nezávislých funkcii tak GraphQL bude určite dobrá voľba.

2.2 Zabezpečenie anonymnej komunikácie

Keďže cieľom je chrániť súkromie užívateľov a do toho sa zahŕňa aj informácia o tom z akej siete bola správa odosielaná, tak bude rozumné použiť sieť TOR - The Onion Router. TOR je systém určený na anonymné prehliadanie webu, riadi prevádzku internetu cez množstvo náhodných serverov, aby utajil používateľovu polohu alebo činnosť pred každým, kto vykonáva dohľad nad sieťou alebo analýzu prevádzky. Používanie TOR sťažuje sledovanie internetovej aktivity, vrátane návštev stránok, postov, IM a iných foriem komunikácie, späť k používateľovi. Je určený na ochranu osobnej slobody používateľov.

Táto sieť na ochanu súkromia užívateľov na základe vrstvového smerovania funguje tak, že pôvodné dáta sú viac krát šifrované a následne sú poslané cez viacero TORstaníc, kde je na každej dešifrovaná jedna vrstva cesty, na ktorú je paket určený. To znižuje pravdepodobnosť, že útočník dešifruje dáta. [21]

Ani táto technika nedokáže zabezpečiť istotu pred odhalením komunikujúcich staníc, ale v priebehu posledných rokov bolo pár bezpečnostných slabín identifikovaných a opravených. Napríklad v roku 2011 výskumníci z francuzkej inžinierskej školy oznámili, že našli spôsob dešifrovania dát v sieti TOR, tento spôsob vyžaduje prístup aspoň k jednej tretine uzlov v sieti. Keďže táto sieť sa stále vylepšuje a vyvíja, takže nebude na škodu, ak sa v systéme vyskytne. Túto technológiu používajú aj torrentové stránky a torrentoví klienti, pomocou ktorých je možné torrenty sťahovať a odosielať [21].

2.3 JSON Web Token

Skratka JWT - (JSON Web Token), jedná sa o internetový štandard na vytváranie údajov s voliteľným podpisom alebo voliteľným šifrovaním. Tokeny sú podpísané a je možné v ňom uschovať dáta. Výhodou je, že je často implementovaný takým spôsobom, že sa generuje a overuje na rovnakom serveri. Z toho vyplýva, že postačuje, aby kľúče k jet tokenu poznal len jeden server a to práve server, ktorý ho generuje a overuje. Token často obsahuje časovú značku, z ktorej je možné overiť, či sa jedná o platný token alebo už expirovaný, vie to predísť zneužitiu odchyteného requestu útočníkom.

Vlastnosti

- má predpísanú štruktúru,
- je zakódovaný do jednoduchého reťazca,
- dokáže v sebe niesť akúkoľvek informáciu, ktorú je možné uložiť ako text,
- je možné overiť jeho pravosť (podpis).

Token sa skladá z troch častí header, payload a signature. Posledná časť je nepovinná a obsahuje podpis celého tokenu. Typ podpisu je možné vybrať z väčšieho množstva algoritmov.

2.4 Firebase

Firebase, je to platforma vyvíjaná spoločnosťou Google na vytváranie mobilných a webových aplikácií. Ponúka množstvo užitočných funkcii, ktoré sú jednoducho implementovateľné do mobilných a webových aplikácií. Pre túto prácu sú podstatnou službou len notifikácie.

Firebase notifikácie, je najpoužívanejšiu službu doručenia push notifikácie pre Android a iOS zariadenia. Do push notifikácie je možné vložiť mnoho nepovinných a nastaviteľných informácii, ktoré sú následne zobrazené na zariadení. Počas chodu aplikácie je možné notifikáciu upraviť, avšak ak aplikácia nie je pri živote, tak nie je možné žiadnym spôsobom notifikáciu upravovať.

2.5 Docker

Docker je v informatike názov pre otvorený softvér, ktorý má za cieľ poskytnúť jednotné rozhranie pre izoláciu aplikácií do kontajnerov v prostredí linux aj windows. Umožňuje to jednoduchý prenos a konfiguráciu serveru zo stroja na stroj. Jediné čo je potreba je mať nainštalovaný docker a spustiť požadovaný image. Za predpokladu, že je docker image správne napísaný, tak vďaka dockeru sa automaticky stiahnu všetky závislostí nutné k chodu daného docker image.

2.6 Vývoj v platforme Flutter

Flutter je novopredstavená platforma od Googlu, umožňuje multiplatformný vývoj na:

- **Android OS** - smartfón, tablet, televízia, auto,
- **iOS** - iPhone, iPad,
- **Fuchsia OS** - čokoľvek, čo bude podporovať (smartfon, tablet, notebook, počítač)
- **web** - webové prehliadače, avšak so správaním veľmi podobným smartfónom.

Keďže táto platforma vyšla v prvej stabilnej verzii 14.6 2019, čo je pomerne nedávno ,nemá rozsiahlu komunitu developerov. Google sa o platformu stará tak, že

vydáva pravidelné aktualizácie a komunikuje s komunitou developerov. Komunita developerov sa každým dňom zväčšuje aj napriek tomu nie je veľa spoločností, ktoré považujú flutter za dostatočne sexy na to, aby v ňom vyvíjali aplikácie. Je to asi hlavne kvôli tomu, že operačný systém Fuchsia je len v betaverzii pre developerov, takže žiadny originálny smartfón alebo notebook nemá daný operačný systém ako pôvodný, takže ešte nie je overený v praxi. Pripravovať aplikácie na trh, ktorý vlastne ešte neexistuje je zrejme veľké riziko aj napriek tomu, že sa jedná o vývoj na štyri platformy v jednom.

Flutter má na vývoj aplikácie úplne iný pohľad než je v klasickom natívnom vývoji a to je ten, že všetky komponenty na obrazovke sú widget a celá aplikácia sa skladá z množstva widgetov, ktoré medzi sebou vhodne komunikujú. Widget je samostatne funkčný prvok na obrazovke. Widget sa môže skladať s ďalších widgetov. Jestvuje pár základných widgetov, ako napríklad widgety na výpis textu, tlačidlá a podobne. Rovnako je nutné sa naučiť nový programovací jazyk zvaný Dart, ktorý je založený na streamovom programovaní. Streamy v programovaní umožňujú reaktívne správanie aplikácie. Pod reaktívnym správaním aplikácie je možné rozumieť, že ak je na dvoch rôznych telefónoch súčasne spustená rovnaká obrazovka rovnakej aplikácie s rovnakými prihlasovacími údajmi, tak zmenou niečoho cez jeden telefón sa zmena automaticky prejaví aj na druhom.

Je nutné preskúmať možnosti kryptografických knižníc a nájsť najefektívnejšiu. Dôležité parametre sú šifrovanie, dešifrovanie a digitálny podpis s čo najvyššou úrovňou bezpečnosti za čo najkratší čas. Šifrovacie postupy a dĺžky kľúčov, ktoré bude potreba v projekte sú: SHA256, AES256 a RSA s 2048 bitovým kľúčom. Po preskúmaní množstva knižníc pre Flutter bola vytvorená tabuľka 2.1 z nej je možné vyčítať, že aktuálne neexistuje žiadna knižnica, ktorá by bola schopná poňať všetky požadované technológie. Samozrejme, by bolo možné projekt postaviť na kombinácií týchto knižníc, ale väčšinu z nich spravili súkromní programátori. Čím viac knižníc je v projekte, tým nestabilnejší produkt je dodaný, lebo sa spolieha na množstvo cudzieho kódu, v ktorom sa skôr či neskôr môže objaviť chyba, ktorú užívateľ knižnice nevie kvalitne odstrániť. Keď nie je knižnica masovo používaná, tak riziko nájdenia chyby pri jej implementácií rastie.

Je tu možnosť napísať si vlastnú knižnicu, v ktorej budú nejaké funkcie prebraté z knižníc z tabuľky avšak väčšinu funkcií bude nutné napísať. Takmer určite nebudú tak efektívne za tak krátky čas a zároveň by bola potreba hardware, ktorý aktuálne nie je k dispozícii. Jedná sa o hardware od spoločnosti Apple. Na vývoj je potrebný počítač a smartfón, ktoré sú od spoločnosti Apple, respektíve na vývoj stačí len smartfón od spoločnosti Apple, ale ak by nastal čas, kedy je nutné vydať aplikáciu

Tab. 2.1: Prieskum kryptografických knižníc pre Flutter

	SHA1	SHA256	AES	RSA	ED25519	ECC
crypto	ano	ano	-	-	-	-
cryptolab	ano	ano	-	-	-	-
simple rsa	-	-	-	ano	-	-
encrypt	-	-	ano	ano	-	-
pointycastle	ano	ano	ano	ano	-	-
steel crypt	ano	ano	ano	ano	-	-
curve25519	-	-	-	-	ano	-
cryptography	ano	ano	-	-	ano	-
viz_dart_ecc	-	-	-	-	ano	-

do Apple store, tak je nutné mať vlastné zariadenie s operačným systémom MAC OS. Vyvinúť slušnú knižnicu pre Flutter, ktorá zahŕňa všetky spomínané funkcie by mohlo byť témou na samostatnú diplomovú prácu.

2.7 Natívny vývoj na Android

Jedná sa o najrozšírenejší vývoj aplikácii na operačný systém Android. Silná komunita užívateľov, mnoho dobre spracovaných dokumentácií a tutoriálov. Množstvo veľkých firiem a šikovných programátorov, ktorí prispeli do open source a vydali mnoho užitočných a efektívnych knižníc, ktoré udržujú. Tieto knižnice nesmierne uľahčujú vývoj, lebo sú odskúšané na množstve aplikácií a mnoho chýb komunita odhalila a opravila. Používa sa programovací jazyk Java Kotlin alebo C++, je možné importovať knižnice, ktoré umožnia reaktívny vývoj. Aktuálne je najnovší operačný systém s označením Android 11.0. Operačný systém Android je spoločnosťou používaný najmenej 10 rokov, je to najpoužívanejší operačný systém na mobilné telefóny.

Pre natívny vývoj prebehol podrobný prieskum možností knižníc zameraných na kryptografiu. Zaujímavé knižnice sú naznačené v 2.2. Knižnica cryptography[22] je priamo od vývojárov Javy, pričom ostatné sú knižnice tretích strán. Napríklad wolfcrypt[23] je od vývojárov wolfSSL, ktorí vytvorili knižnice pre android natívne v C++. Knižnice Spongy castle[24] a JPBC[25] sú v Jave. Wolfcrypt má mnoho funkcií, dokonca podporuje aj eliptické krivky. Keďže v návrhu logiky aplikácie sa nespoliehame priamo na DH, ale na AES - Advanced Encryption Standard - Štandardizovaný algoritmus pre Symetrickú šifru), že sa jedná o symetrickú šifru nie je veľké mínus, lebo kľúč stačí zdistribuovať raz a na to postačí jednorázovo dlhotrvajúce RSA, tento proces je podrobne opisovaný dalej v práci 4.3.4.

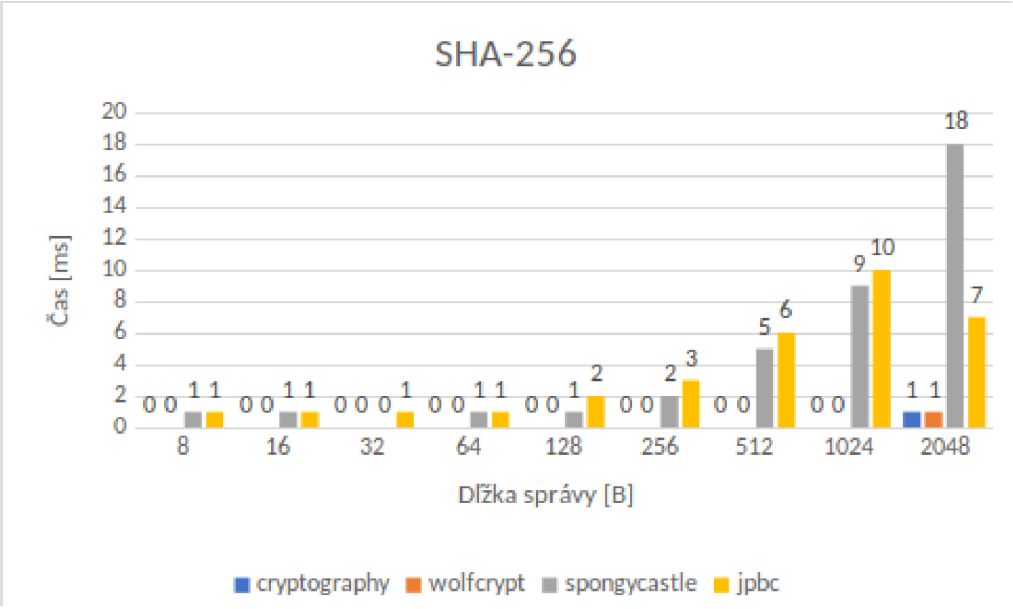
Tab. 2.2: Prieskum kryptografických knižníc pre Android

	SHA1	SHA256	AES-256	RSA-2048	ED25519	ECC
cryptography	ano	ano	ano	ano	-	-
wolfcrypt	ano	ano	ano	ano	ano	ano
spongy castle	ano	ano	ano	ano	ano	ano
JPBC	ano	ano	-	-	-	ano

2.7.1 Testovanie kryptoknižníc

Testovanie kryptoknižníc pre algoritmy SHA-256 a AES-256. Sledovaný bol čas vykonávania funkcie pre SHA-256, je to tvorba hashu a pre AES-256 sa sledoval čas šifrovania a dešifrovania správy, pričom menená bola práve dĺžka správy. Testované dĺžky správ sú: 8 B, 16 B, 32 B, 64 B, 128 B, 256 B, 512 B, 1024 B a 2048 B.

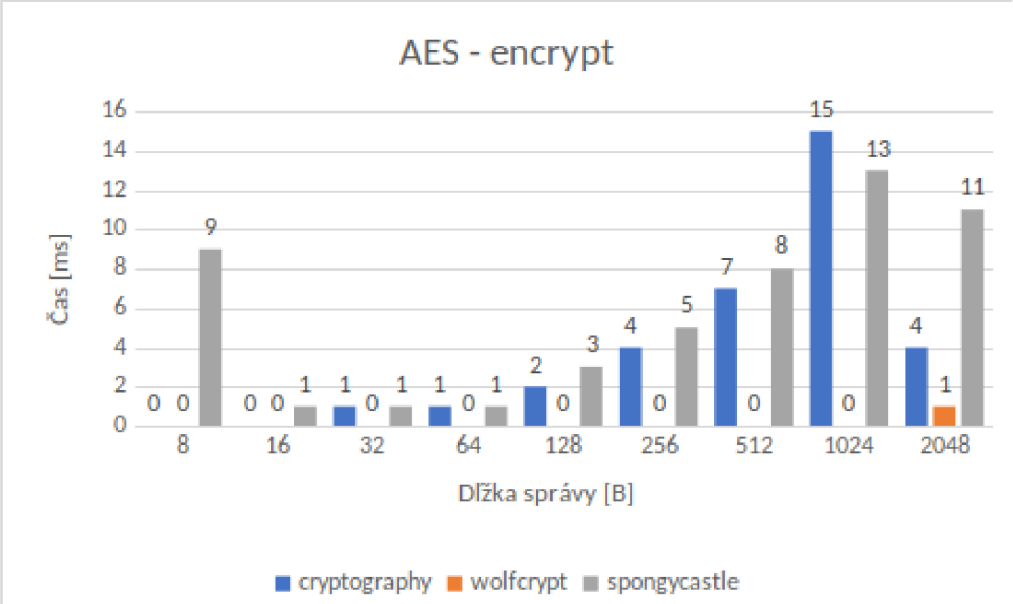
Výsledky sú vyobrazené viz obr. 2.1, 2.2. V grafe sa nachádzajú nuly, je to spôsobené tým, že aj keď meranie bolo v ms, tak namerané hodnoty boli hodnoty s časom 0, znamenajú že priemerný čas vykonania funkcie trval menej, než 1 ms. Funkcia daného algoritmu sa pustila 1000 krát za sebou a zaznamenávalo sa trvanie. Priemerný čas sa vypočítal vydelením celkového času číslom 1000. Nuly su pri hodnotách, s až moc nízkou dĺžkou správ pre náš systém, preto sa kladie dôraz na časy udávané pri dlhších správach.



Obr. 2.1: SHA-256 benchmark knižníc na základe dĺžky vstupu

Z obr. 2.1 je možné vidieť, že funkcia SHA-256 je pomerne jednoduchá funkcia,

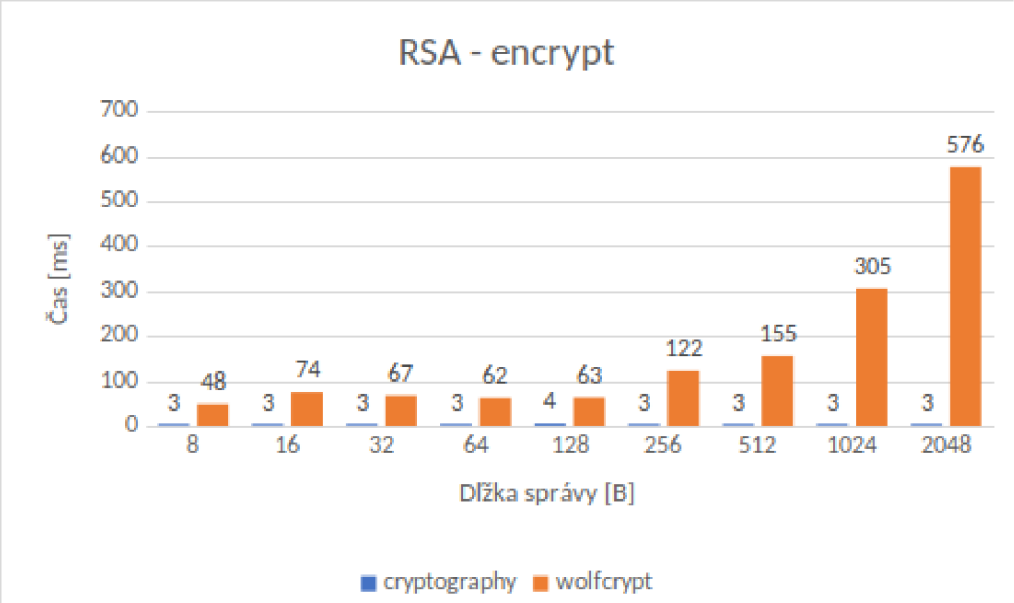
pre väčšinu knižníc netrvá príliš dlho. Pre knižnice SpongyCastle a JPBC sa pri dlžke správy 512 B citeľne predlžuje čas vykonania funcie. Užívateľ pri používaní telefónu nerád čaká a čas 3-5 sekúnd je už na hrane pocitu sekania aplikácie. Jedná sa síce o funkciu, ktorá robí hash, takže užívateľ nemusí čakať na vykonanie funkcie, lebo bude prebiehať po odoslaní správy adresátovi, no správa bude musieť prejsť cez sekvenciu kryptoalgoritmov, kde je treba ušetriť čas. Ideál by bol, keby čas od odoslania správy a zašifrovania do prijatia správy adresátom a jej dešifrovania prešiel čo najkratší čas. Ideálne by bolo, keby výstupná aplikácia tejto práce bola schopná doručiť správu do 6-10 sekúnd. Po optimalizácii requestov by bolo možné dosiahnuť čas pod 5 sekúnd čo je konkurencieschopný čas.



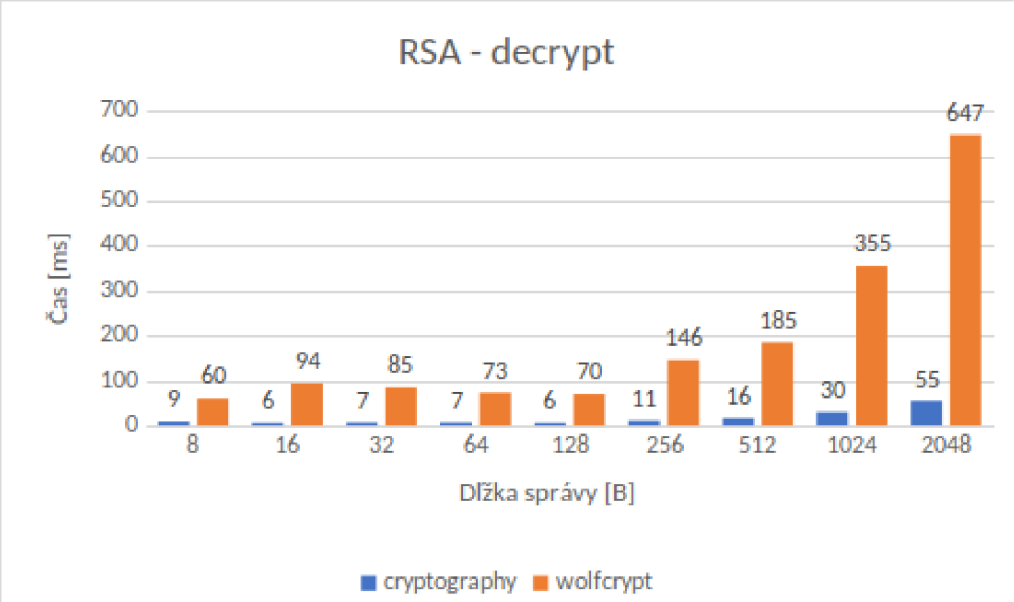
Obr. 2.2: AES encrypt benchmark knižníc na základe dĺžky vstupu

Porovnanie algoritmu AES pozostávalo z dvoch krokov. Bol overovaný čas šifrovania správy a dešifrovania správy, keďže oba tieto úkony budú prebiehať na smartfónoch tak je nutné dbať na čas šifrovania aj dešifrovania. Z obr. 2.2 je možné vidieť, že knižnica Wolfcrypt pre správy dlhšie než 512 B naberá veľké časy. Ako dvojica najvhodnejších knižníc sa javí knižnica Crypography od Javy a Wolfcrypt písaný v C++. Ak počítame s tým, že priemerná správa má tak 100 znakov s kódovaním UTF-8 priemerne na znak 2 B, je to 200 B, takže najväčší dôraz je dávaný na správy do 256 B prípadne 512 B. Pre dosiahnutie dĺžky 2048 B je nutné napísať priemerne 2000 znakov, čo sa vymyká z priemerne dlhej správy, takže predpoklad je, že pri takto dlhej správe bude užívateľ tolerantný a zvládne čakať kúsok dlhší čas.

Na obr. 2.3 a 2.4 je možné vidieť porovnanie času potrebného na šifrovanie a dešifrovaním dvoch knižníc (Cryptography a Wolfcrypt). Pričom z grafov je očividné,



Obr. 2.3: RSA-2048 encrypt benchmark knižníc na základe dĺžky vstupu

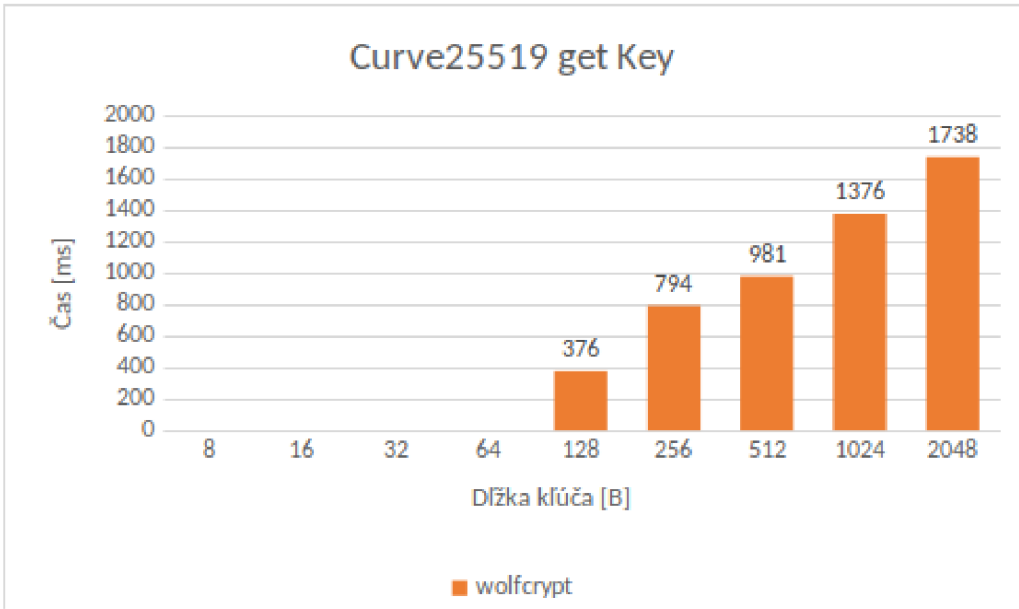


Obr. 2.4: RSA-2048 decrypt benchmark knižníc na základe dĺžky vstupu

že knižnica Cryptography potrebuje omnoho menší čas na šifrovaníe aj dešifrovanie správ a to aj napriek nutnosti rozložiť správy väčšie než 126 B na viacero subkryptogramov s dĺžkou 128 B. Ako absolútny víťaz v kategórii algoritmu RSA je označená knižnica cyptography, keďže dosahuje časy, ktoré sú ledva 10% z času, ktorý bol zmeraný u knižnice wolfcrypt, rovnako zdanlivá funkcia závislosti času strávenom

na šifrovaní/dešifrovaní správy na dĺžke vstupu je omnoho priaznivejšia pre väčšie vstupy než spomínaná knižnica wolfcrypt.

Relatívne novinkou v kryptoknižniciach sú eliptické krivky. Po dôkladnejšom preskúmaní zabezpečenia jednotlivých IM aplikácií bolo zistené, že mnoho z nich používa eliptické krivky, konkrétne Curve25519. Curve25519 je štandardizovaná eliptická krivka, ktorá sa používa na výmenu kľúčov Diffie-Hellmanovmu protokolu. Krivka je definoaná nad telesom $GF(2^{255} - 19)$, pričom $2^{255} - 19$ je prvočíslo, podľa ktorého je krivka pomenovaná. Prebehlo meranie, ktorého úlohou bolo zistiť čas strávený na získanie kľúča pomocou Curve25519 na základe dĺžky kľúča. Výstupom merania je obr. 2.5, kde je znázornená závislosť času na dĺžke požadovaného kľúča. Podarilo sa rozbehnúť len Wolfcrypt knižnicu, avšak zatiaľ to na demonštráciu postačí, keďže je potreba zistiť, či je efektívnejšie používať RSA alebo spomínanú eliptickú krivku. Výsledkom je, že pri RSA aj napriek väčšej dĺžke kľúča je čas strávený šifráciou/dešifráciou kratší. Eliptické krivky samotné nie sú v kryptografii novinkou, ale na smartfónoch sa používajú pomerne krátko a smartfóny nie sú na ne stavané a optimalizované.



Obr. 2.5: Benchmark na získanie kľúča pomocou eliptickej krivky Curve25519

2.7.2 Zvolené technológie, knižnice a kryptosystémy

Ako technológie, pomocou ktorých bude splnené zadanie tejto páce boli vybraté na základe predchádzajúcich testov nasledovné.

- Vývoj len na operačný systém Android pomocou Android studio s programovacím jazykom Kotlin
- kryptografická knižnica Andoid cryptography
- GRPC - diaľkové volanie procedúr od Googlu, umožňuje reaktívny prístup komunikácie serveru s aplikáciou
- PostgreSQL - voľne šíriteľný objektovo-relačný databázový systém
- Programovací jazyk na serveri Golang a príležitostne Python
- kryptografické algoritmy
  - hashovacia funkcia - SHA-256 pre integritu dát
  - symetrická šifra - AES-256 pre šifrovanie dát pred serverom
  - RSA - asymetrická šifra - na distribúciu symetrických kľúčov

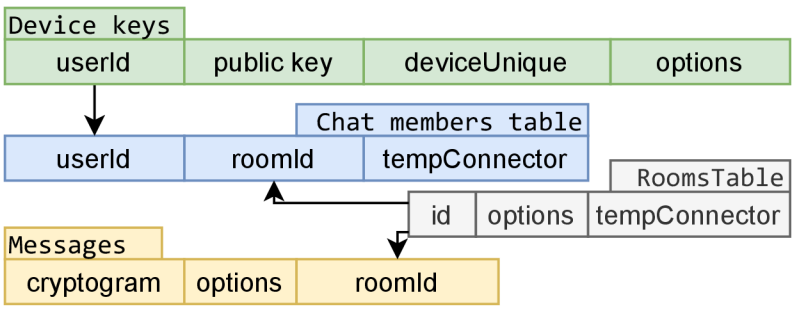
3 Základný popis architektúry

Navrhnutý systém podporuje mnoho ďalších vlastností, ktoré umožňujú dosiahnuť vyššieho súkromia užívateľov a dávaju užívateľom väčšiu kontolu nad ich dátami, napríklad:

- nastavenie životnosti správ po ktorej sa úplne odstránia zo systému aj keď je kľúč platný,
- limitovanie informácii poskytnutých oponentovi (napr. komunikovanie pod dočasným pseudonymom),
- žiadosť o povolenie vykonať screenshot obrazovky,
- nastavenie životnosti všetkým typom správ (počet otvorení, možnosť otvorenia na celú obrazovku),
- odoslanie informácie o prečítaní správy oponentom.

Navrhnuté riešenie umožňuje užívateľom využívať vlastný server so správami a réžiou miestnosti, na ktorý sa aplikácia pripojí len jednoduchou zmenou endpointov. Všetky servery okrem autorizačného je možné naklonovať a rozbehnúť vlastný server aplikácie. Vďaka tomu je možné pri prvom spustení aplikácie určiť, ku ktorému serveru sa bude zariadenie pripájať, de facto na ktorom serveri budú uložené užívateľove kľúče. Ten, kto si stiahne, nakonfiguruje a spustí servre vytvorené v tejto práci, si po prihlásení do aplikácie vpíše cestu k svojim servrom.

Navrhnutý IM systém je založený na inovatívnom návrhu databázových tabuliek. Základom je tabuľka miestností, v ktorej je najdôležitejším identifikátorom ID miestnosti a dalšie konfiguračné dáta. Ďalšia tabuľka obsahuje informáciu, aký užívateľ (`userId`) sa nachádza v ktorej miestosti (`roomId`). Posledná najdôležitejšia tabuľka obsahuje správy, kde každý riadok obsahuje šifrovanú správu, identifikátor miestností do ktorej správa patrí a ID užívateľa, ktorý správu odoslal plus ďalšie konfiguračné dáta. Základná architektúra tabuľkového systému a ich závisostí je zobrazená na obrázku 3.1. Návrh je multifunkčný, ale aktuálne je implementovaný len na android OS.



Obr. 3.1: Základná architekrúra tabuľkového systému.

Vďaka návrhu distribúcie kľúčov, je každá správa šifrovaná symetrickým kľúčom a uložená na server do už spomínanej tabuľky. Správny kľúč k danej správe majú len jej adresáti a odosielateľ. A do miestnosti sa nedostane užívateľ, ktorý do nej nebol pozvaný a ak by sa tam nejakým spôsobom dostal alebo by nejakým spôsobom dokázal zistiť, ktoré správy patria do miestnosti na ktorú útočí, tak bude vidieť len kryptogramy správ ku ktorým nemá kľúč. Kľúč má len zariadenie, ktoré bolo do miestnosti pozvané z vnútra.

4 Návrh kryptosystému pre IM aplikáciu

Cieľom navrhovaného kryptosystému je umožniť verejný prístup do akejkoľvek DB - Databáza na serveri. Keďže sa predpokladá otvorený prístup do DB, tak treba všetko ukladať v šifrovanej forme, tak aby nebolo možné zo zašifrovaných dát zistiť

- odosielateľa správy,
- adresáta,
- obsah správy,
- metadata,
- informácie o zariadení, z ktorého bola správa odoslaná - IP, MAC, poloha a iné metadata o zariadení.

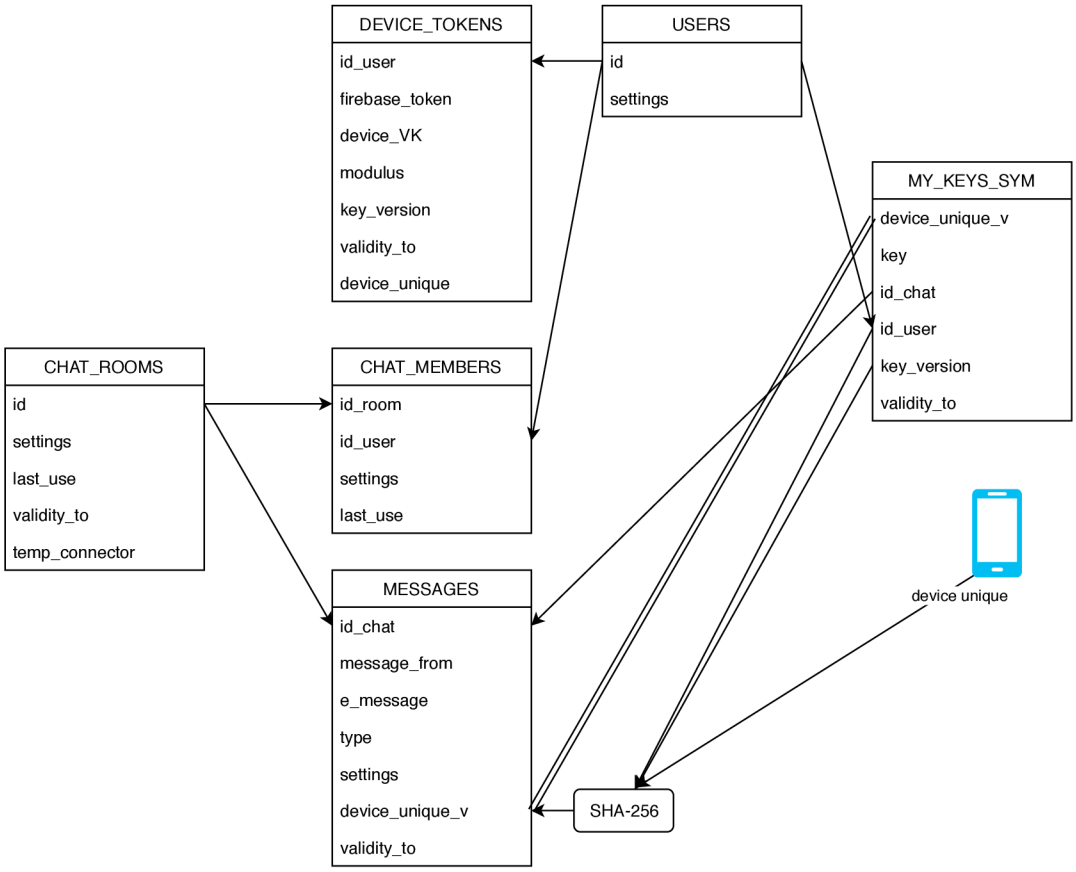
4.1 Návrh databáz

Na chod systému bude potrebná PostgreSQL databáza, v ktorej je nutné zadefinovať tabuľky a vzťahy medzi nimi. V tejto kapitole sú opísané tabuľky potrebné na zaistenie požiadaviek. Vzťahy medzi jednotlivými tabuľkami sú znázornené na obr. 4.1, jednotlivým tabuľkám je venovaná pozornosť ďalej v texte.

4.1.1 Server

Na strane servera je treba najmenej nasledujúcich päť tabuliek

- **USERS** - tabuľka všetkých užívateľov obsahuje
  - `id` - unikátne id užívateľa,
  - `settings` - nastavenia užívateľa (napr predvolené zverejnovanie/nezverejnovanie mena alebo prezývky, po akej dobe neaktivity zmazať účet),
- **DEVICE_TOKENS** - tabuľka potrebných informácii na prepojenie užívateľa a jeho zariadení obsahuje
  - `userId` - hash id užívateľa z tabuľky **USERS** a tajného kľúča, aby nebolo možné dohľadať užívateľa
  - `firebase token` - token platformy Firabase, ktorý umožňuje doručenie notifikácie na užívateľov telefón, keď aplikácia nie je na jeho zariadení spustená, ale nainštalovaná
  - `deviceVK` - Verejný RSA kľúč asymetrickej kryptografie (jedná sa o VK s SK, ktorý je uložený v zariadení)
  - `verzia páru kľúčov` (užívateľ si kedykoľvek môže vygenerovať nové alebo zmazať staré)
  - `validityTo` - platnosť vygenerovaných kľúčov



Obr. 4.1: Vzťahy medzi jednotlivými tabuľkami

  - `device_unique` - unikátny identifikátor zariadenia, aby bolo možné jednoducho odstrániť zariadenie zo zariadení autoriovaných čítať správy.
- **CHAT_ROOMS** - tabuľka obsahujúca všetky aktívne chatové miestnosti obsahuje
  - `id` - unikátne id miestnosti
  - `settings` - nastavenia miestnosti
  - `lastUse` - posledná aktivita v miestnosti
  - `validity` - čas, do ktorého ak nedôjde k predĺženiu platnosti chatu, tak bude celá miestnosť úplne odstránená zo systému
  - `tempConnector` - unikátny identifikátor pre chat, pomocou ktorého sa pripojí nový člen. Po pripojení správneho počtu užívateľov (settings) sa temp identifikátor nastaví na null, ďalšia možnosť nastavenia na null je, ak je v settings obmedzená doba na pripojenie nových užívateľov.
- **CHAT_MEMBERS** - tabuľka obsahujúca všetkých aktívnych užívateľov obsahje
  - `chatId` - unikátne id chatovej miestnosti,

  - `userId` - hash id užívateľa z tabuľky USERS a tajného kľúča, aby nebolo možné dohľadať užívateľa,
  - `settings` - nastavenia člena v miestnosti,
  - `lastUse` - časť poslednej aktivity užívateľa v miestnosti.
- **MESSAGES** - tabuľka obsahujúca všetky správy užívateľov,
  - `chatId` - unikátne id chatovej miestnosti,
  - `message_from` - hash id užívateľa z tabuľky USERS a tajného kľúča, aby nebolo možné dohľadať užívateľa,
  - `message` - zašifrovaný obsah správy,
  - `type` - typ správy text/obrázok/gif/súbor,
  - `settings` - nastavenia správy, napr v prípade obrázku
    * možnosť otvoriť na celú obrazovku
    * možnosť urobiť screenshot
    * možnosť stiahnutia
    * maximálny počet otvorení na celú obrazovku
    * či obsahuje náhľad alebo nie (či je viditeľný obrázok v chate alebo až po otvorení na celú obrazovku),
  - device_unique_v - hash údajov z databáz **CHAT_KEYS_SYM** prípadne **MY_KEYS_SYM** v telefóne. Konkrétne hash device_unique + userId + key_version
  - najkratšia platnosť spomedzi (kvôli hierarchickému mazaniu správ)
    * **CHAT_ROOMS**.VALIDITY
    * **DEVICE_TOKENS**.VALIDITY
    * **USERS**.VALIDITY

4.1.2 Smartfón

Potrebné tabuľky na každom zariadení, ktoré je do systému pripojené, spravidla smartfón. V zariadení budú nasledovné tabuľky

- **MY_KEYS_ASYM** -> súkromné RSA kľúče asymetrického kryptosystému pre dané zariadenie
  - SK -> SK k VK z tabuľky **DEVICE_TOKENS**
  - validity -> platnosť súkromného kľúča
- **CHAT_KEYS_SYM** -> kľúče AES symetrického kryptosystému oponenta pre každú chatovú miestnosť, v ktorej sa užívateľ nachádza
  - device_unique_v -> hash údajov z databáz **CHAT_KEYS_SYM** prípadne **MY_KEYS_SYM**
  - key -> kľúč symetrického kryptosystému
  - chatId -> id chatovej miestnosti

  - userId -> id užívateľa z **DEVICE_TOKENS**.USER_ID
  - key_version -> indikátor poradia key – ak **DEVICE_SYM_KEY** obsahuje pre dané device_unique_v väčšie key_version než je najväčšie v telefóne, tak sa z **DEVICE_SYM_KEY** stiahne a dešifruje novší kľúč a pridá sa mu rovnaká verzia ako je v **DEVICE_SYM_KEY**.KEY_VERSION
  - validity -> čas platnosti z **DEVICE_TOKENS**.VALIDITY_TO

  Všetky správy prichádzajúce do telefónu sú dešifrované kľúčom key, ktorý sa nájde podľa identifikátoru (hash DEVICE_UNIQUE_VALUE + USER_ID + KEY_VERSION) v tabuľke **CHAT_KEYS_SYM**
- **MY_KEYS_SYM** -> vlastné kľúče symetrického kryptosystému pre každú chatovú miestnosť, v ktorej sa užívateľ nachádza
  - device_unique_v -> hash údajov z databáz **CHAT_KEYS_SYM** prípadne **MY_KEYS_SYM** Konkrétne hash device_unique + my_user_id + key_version
  - key -> kľúč symetrického kryptosystému
  - chat_id -> id miestnosti, ku ktorej kľúč patrí
  - userId -> id užívateľa prihláseného v zariadení (táto položka je tu len na zjednodušenie prístupu a overenie funkčnosti, odstrániť tento parameter z tabuľky bude prvá vlna optimalizácie)
  - key_version -> indikátor poradia key – nejaký čas pred vypršaním kľúča je možné ho predĺžiť/vygenerovať nový, ak sa vygeneruje nový, zvýši sa key_version oproti minulej
  - validity -> čas platnosti z **DEVICE_TOKENS**.VALIDITY_TO

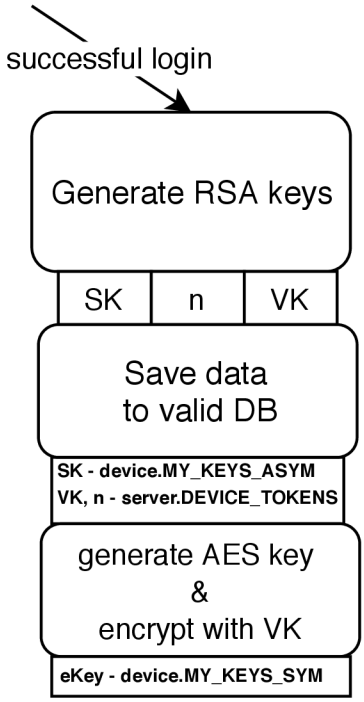
  Všetky správy opúšťajúce telefón sú šifrované najnovším
  KEY (najvyššie key_version) z tabuľky **MY_KEYS_SYM**

4.2 Prepojenie tabuliek

Keďže tabuliek je pomerne dosť, tak aj komunikácia a previazanosť medzi nimi je komplikovaná. Schválne je to podelené na mnoho tabuliek a to z toho dôvodu, aby bolo možné zlepšiť výkon komunikácie. Keď je viacero vhodne navrhnutých tabuliek, tak je práca s nimi jednoduchá a rýchla, ak príde čas na optimalizáciu tak stačí upravovať len miesta, kde sa dané tabuľky spájajú. Optimalizácia typu dlhotrvajúce odpovede treba rozdeliť na viacero čiastkových, tak aby sa užívateľovi zobrazoval obsah postupne a nie celý až po čase najdlhšie trvajúceho bloku.

4.2.1 Distribúcia kľúčov

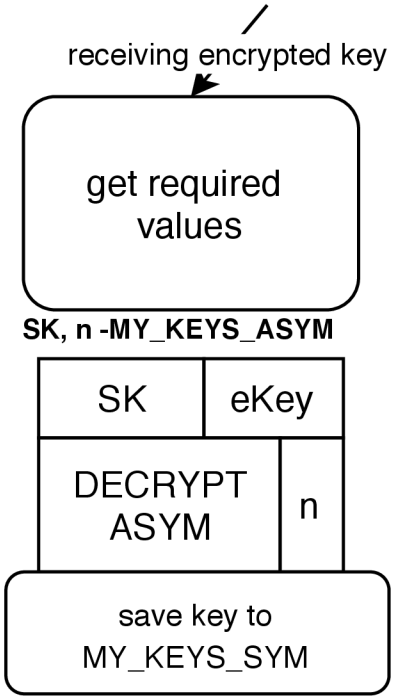
Prepojenie tabuliek je komplikovaný a veľký diagram, preto je rozdelený na viacero logických blokov, ktoré budú vysvetlené nižšie. Teraz nebude riešený proces registrácie a prihlásenia ale distribúcia dôležitých kľúčov a logika šifrovania a dešifrovania. Je predpokladané, že na oboch smartfónoch medzi ktorými chce byť zinicializovaá komunikácia majú nainštalovanú aplikáciu, ktorá implementuje logiku, ktorá je opisované v tejto práci. Na smartfónoch sú prihlásení rôzni užívatelia, ktorí sú uložení v tabuľke *USERS* a každému je pridelené unikátne id užívateľa. Následná komunikácia z Obr. 4.2 bude prebiehať po prihlásení užívateľa na smartfón.



Obr. 4.2: Distribúcia a generovanie kľúčov po prihlásení do aplikácie

Po úspešnom prihlásení užívateľa a vytvorení chatovacej miestnosti sa na serveri alebo smartfóne vygeneruje pár RSA kľúčov, z čoho sa údaje VK, SK a modulus uložia do potrebných databáz. Uloženie čísla n a VK do DB.DEVICE_TOKENS je jednoduché. Na ozrejmenie procesu dešifrácie a uloženia symetrického kľúča je uvedený Obr. 4.3, kde je naznačený proces dešifrácie a získanie parametrov potrebných pre úspešnú dešifráciu a uloženie kľúča. Po dešifrácii kľúča sa uloží ako nový riadok do tabuľky MY_KEYS_SYM. Táto tabuľka obsahuje viacero informácií, ktoré aktuálne nie sú potrebné.

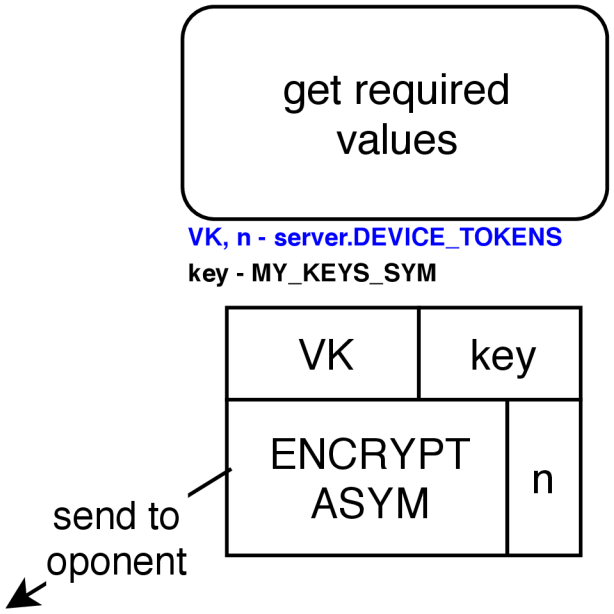
Po úspešnom uložení všetkého potrebného do príslušných tabuliek, je zaujímavá tabuľka CHAT_ROOMS a CHAT_MEMBERS. V tabuľke miestností je vytvorená



Obr. 4.3: Dešifrovanie zašifrovaného symetrického kľúča

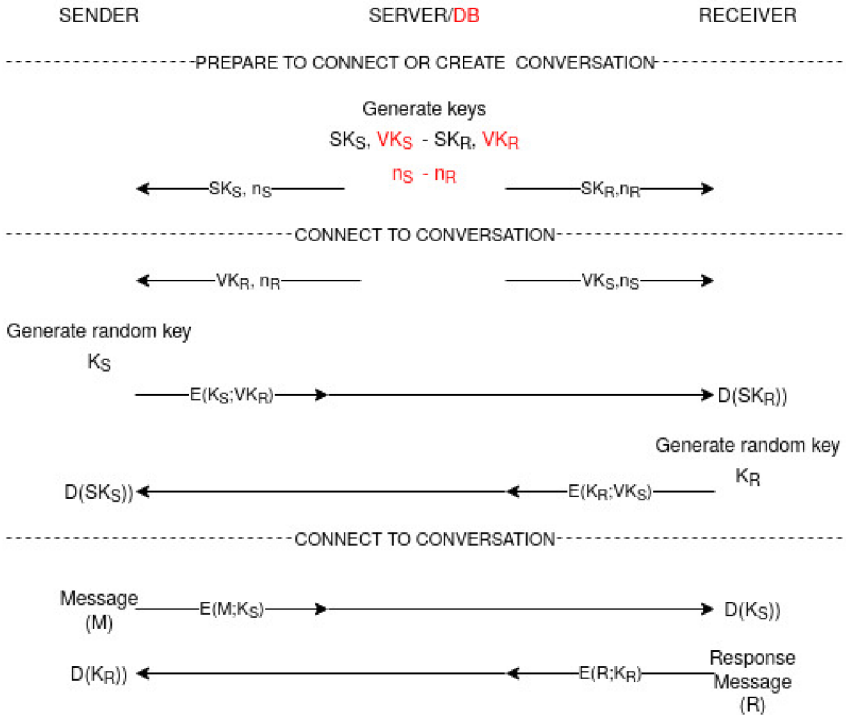
miestnosť. Tabuľka členov ešte neobsahuje dvojicu užívateľov, ale len jedného, toho ktorý miestnosť vytvoril. Pre úspešné vytvorenie chatovej miestnoti musí iniciátor cez nejaký kanál oznámiť oponentovi unikátny identifkátor na prepojenie s jeho chatovou miestnosťou, ktorú oponent vyplní a ak sa bude zhodovať s hodnotou v tabuľke CHAT_ROOMS.tempConnector, tak sa táto hodnota z tabuľky zmaže (nebude možné pridať viacero členov) a užívateľ sa pridá do tabulky členov. Na oznámenie unikátneho identifikátora miestnosti, je možné použiť odoslanie SMS, telefonát alebo akúkoľvek inú IM platformu. Tento užívateľ absolvuje proces generovania a ukladania kľúčov ako je znázornené na Obr. 4.2 a 4.4. Nasleduje proces výmeny kľúčov, ktorý je vyobrazený na Obr.4.4

Na Obr. 4.6 je znázorné zašifrovanie symetrického kľúča pomocou verejného kľúča oponenta. Proces zistenia kto je oponent je založený na znalosti identifikátora miestnosti, v ktorej sa užívateľ nachádza. Z nej si užívateľ vie vyzistiť id oponenta a následne v tabuľke DEVICE_TOKENS vyhľadať toho užívateľa, s ktorým komunikuje a zašifrovať kľúč symetrického kyrptografického systému VK oponenta. Po úspešnom odoslaní a doručení oponentovi nasleduje rovnaký proces ako je v Obr. 4.3 s tým rozdieľom, že kľúč je uložený do tabuľky CHAT_KEYS_SYM. Tento proces prebieha ešte raz opačným smerom, t.j od zariadenia, ktoré práve prijalo symetrický kľúč oponenta k odosielateľovi kryptogramu, ktorého obsahom bol kľúč. Spomínaná



Obr. 4.4: Zašifrovanie symetrického kľúča pomocou VK oponenta a následné zaslanie

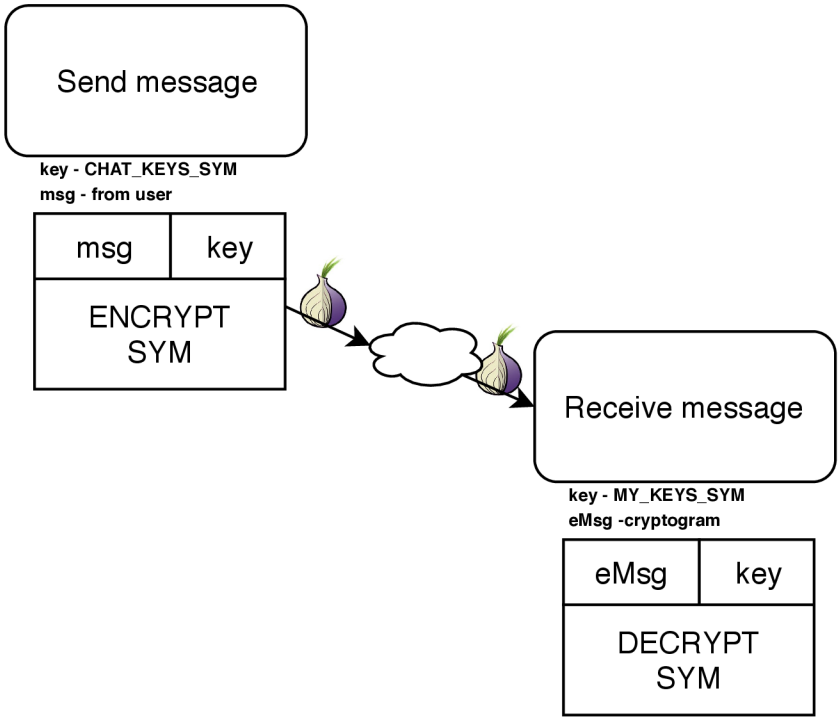
logika je schématicky v zjednodušenej forme opísaná na Obr. 4.5.



Obr. 4.5: Zjednodušený schématický popis distribúcie správ

Po bezchybnom prejdení spomínaného procesu je všetko pripravené na komuni-

káciu, ktorá bude prebiehať ako je znázornené na Obr. 4.6. Stredný stĺpec prezentuje server a zároveň databázu. Do databázy je totiž nutné uložiť verejné kľúče všetkých užívateľov a to z dôvodu, že konverzácia môže byť vytvorená aj za stavu, kedy aspoň jeden zo zúčastnených je offline. Niektoré spojenia sa skladajú z dvoch, napríklad distribúcia generovaného kľúča, je to preto, lebo tento kryptogram je uložený v databáze z rovnakého dôvodu ako verejné kľúče.



Obr. 4.6: Zašifrovanie správy u odosielateľa a dešifrovanie u adresáta

Na diagrame je znázornený príklad komunikácie medzi dvoma účastníkmi, pričom komunikácia neprebieha priamo so zariadením oponenta, ale prechádza to cez server, ktorý to ukladá do DB.MESSAGES a to z dôvodu, že je v pláne podporovať komunikáciu, kde budú viac než dvaja účastníci. Komunikácia medzi klientom a serverom bude prechádzať TOR sieťou, ktorá funguje tak ako je opisované v kapitole 2.2. Keďže aj smerovanie cez TOR sieť trvá nejaký čas, tak je treba všetko nastaviť, tak aby bol splnený cieľ. Ako bolo spomínané v kapitole 2.7.1 cieľom je, aby komunikácia z Obr. 4.6 trvala čo možno najkratší čas, prijateľný čas bude v rozmedzí 5-10s.

4.3 Flow aplikácie

Následne je potreba detailnejšie priblížiť celkový dátový tok aplikácie od registrácie až po odosielanie a prijímanie správ.

4.3.1 Registrácia užívateľa

Aj napriek tomu, že sa jedná o chatovú aplikáciu na ochranu súkromia, tak je od užívateľov vyžadovaná registrácia. Užívateľov je nutné registrovať, aby každý mal prístup k svojim správam. To sa dá docieliť len autentizáciou užívateľa na základe prihlásenia / registrovania.

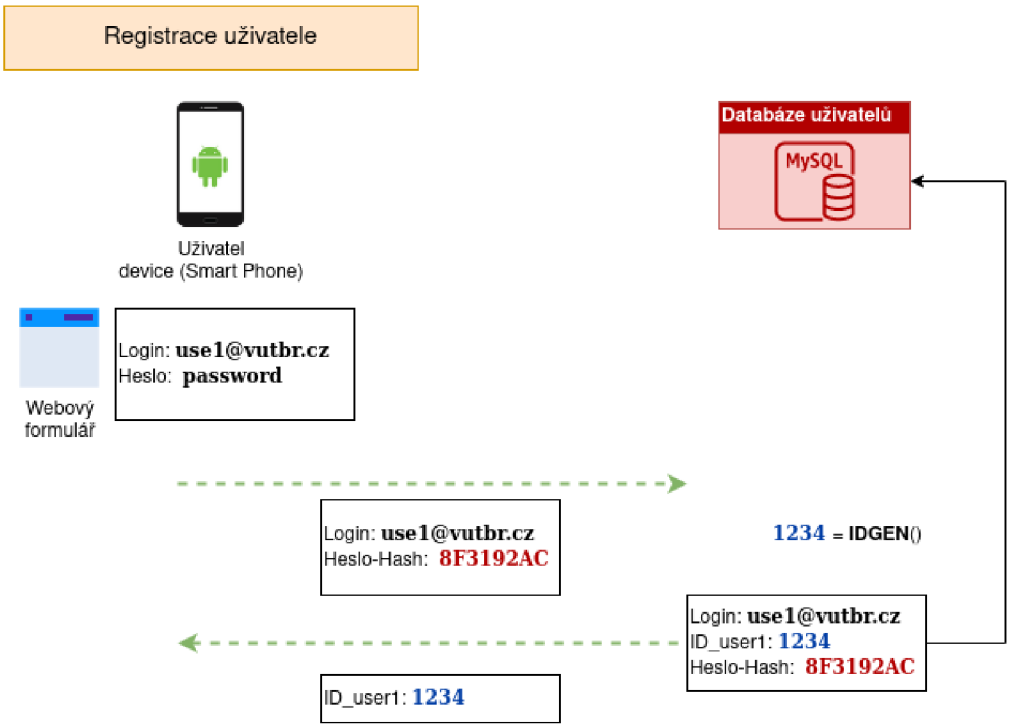
Na úspešnú registráciu užívateľa je nutné zvoliť login, štandardne sa jedná o emailovú adresu alebo telefónne číslo, ktorých autentičnosť je následne overená. Dodatočné overenie emailu a telefónneho čísla slúži na kontrolu, či daný údaj reálne existuje a zároveň či patrí užívateľovi, ktorý sa ním registruje, aby nebolo možné založiť a používať účet, ktorý je spojený s cudzím údajom a zároveň zaručuje, že daný login je v systéme unikátny.

Keď užívateľ vyplní prihlasovacie meno, pod ktorým sa bude prihlasovať, je nutné udať heslo. Užívateľské meno a hash hesla sa odošle na server. Server skontroluje, či sa daný login v systéme nenachádza a ak nie, tak vloží nového užívateľa do tabuľky USERS. Po úspešnom vložení je zariadeniu, z ktorého sa užívateľ registroval navrátený unikátny identifikátor užívateľa v systéme takzvané *userId*. Celý tento postup je znázornený na Obr. 4.7

4.3.2 Registrácia zariadenia

Každé aktívne zariadenie v systéme musí mať platnú dvojicu RSA kľúčov, ktoré sa generujú hneď po úspešnej autorizácii užívateľa. Systém potrebuje mať k dispozícii ukinátny identifikátor každého zariadenia, ktorý sa nemení v čase ani po reinštalácii aplikácie. V ideálnom prípade sa používa MAC (media access control address) adresa zariadenia, ale tá v Android zariadeniach nie je pre programátora prístupná od verzie Android 6.0. Je možné použiť androidID 4.3.2 ,čo by mal mať podobnú funkcionalitu ako MAC adresa, ale nie tak úplne. V dokumentácii je písané, že nie je zaručené, že vždy sa bude jednať o unikátnu sekvenciu bitov a to z dôvodu, že nie všetci výrobcovia dané ID nastavia správne pre všetky ich modely. Z toho vyplýva, že je možné, že niektoré typy/rady telefónov, budú vracať rovnaké androidID a z toho dôvodu budú musieť byť tieto zariadenia zo systému vylúčené.

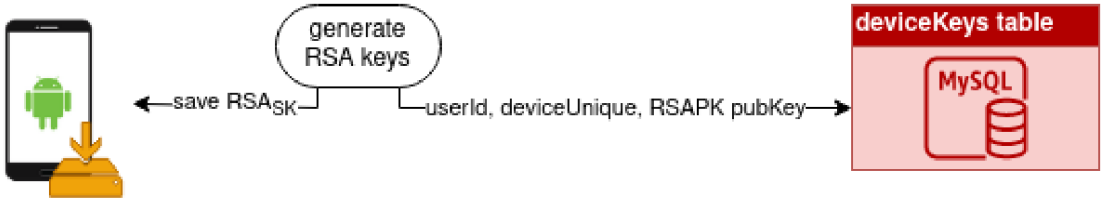
```
deviceUnique = Settings.Secure.getString(
                        contentResolver,
                        Settings.Secure.ANDROID_ID
```



Obr. 4.7: Registrácia užívateľa

)

Spomínaný unikátny identifikátor vygenerovaný pomocou 4.3.2 (7P6SQUA7B) je v systéme nazvaný ako deviceUnique. Na obrázku 4.8 je zobrazené flow, ktoré nastáva po úspešnom prihlásení do aplikácie, zistí sa identifikátor deviceUnique a na strane telefónu sa vygeneruje dvojica RSA kľúčov. Súkromný kľúč je uložený do lokálnej databáze v zariadení a verejný kľúč je spolu s userId a deviceUnique poslaný na server do tabuľky device_keys. Spolu s údajmi sa na server odošle firebaseToken, ktorý je významný pri odosielaní notifikácií, ktoré sú preberané v 5.6 neskôr.

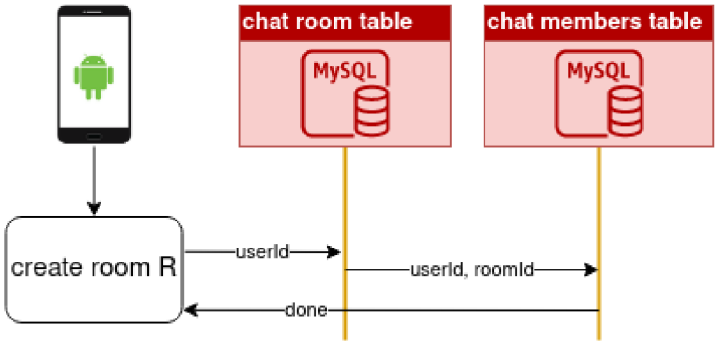


Obr. 4.8: Registrácia zariadenia

4.3.3 Réžia miestností

Po úspešnej registrácii užívateľa a jeho zariadenia nie je užívateľ schopný s nikým komunikovať. Na to aby bolo možné odoslať niekomu správu tak je nutné vytvoriť chatovú miestnosť a do nej prizvať oponenta.

Vytvorenie miestnosti prebieha tak, ako je naznačené na Obr. 4.9. Na vytvorenie miestnosti je potreba iba userID. Telefón pošle na server userId prihláseného užívateľa, server vytvorí záznam v tabuľke **CHAT_ROOM**, po úspešnom vložení záznamu, sa zistí id novo-vytvorenej miestnosti (roomId), ktoré je spolu s userId uložené v tabuľke **CHAT_MEMBERS**.



Obr. 4.9: Vytvorenie miestnosti

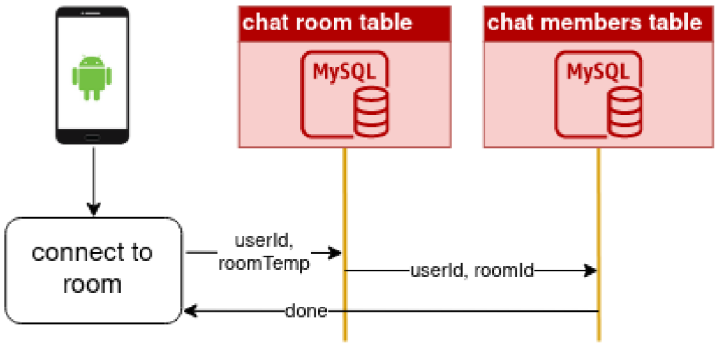
V takto vytvorenej miestnosti je zatiaľ len užívateľ, ktorý ju vytvoril, avšak ten má k dispozícii automaticky generovaný kód zvaný temp_connector. Pomocou tempConnectoru je možné do miestnosti prizvať oponenta. *Tento kód je nutné doručiť oponentovi iným komunikačným kanálom, napríklad SMS, email, messenger*

Oponent sa do miestnosti pripojí podobným spôsobom, tento postup je naznačený na Obr. 4.10. Užívateľ do dialogu vloží tempConnector a v zápätí sa na server odošle request obsahujúci userId a tempConnector. Ak je nájdená v tabuľke **CHAT_ROOM** zhoda v temp_conenctore, tak sa id miestnosti a žiadateľa o prijatie do miestnosti uloží do tabuľky **CHAT_MEMBERS**. Akonáhle je tempConnector použitý tak sa maže čo spôsobí, že už je neplatný.

4.3.4 Distribúcia kľúčov

Aby bola zabezpečená funkčnosť systému, založená na šifrovaní typu end-to-end, tak je nutné dopraviť kľúč, ktorým sa šifrujú správy adresátovi, ktorý ich dešifruje. Systém je navrhnutý tak, že:

- každá správa je šifrovaná symetrickým kľúčom,
- každá správa je dešifrovaná rovnakým kľúčom, ktorým bola zašifrovaná,



Obr. 4.10: Pripojenie k miestnosti

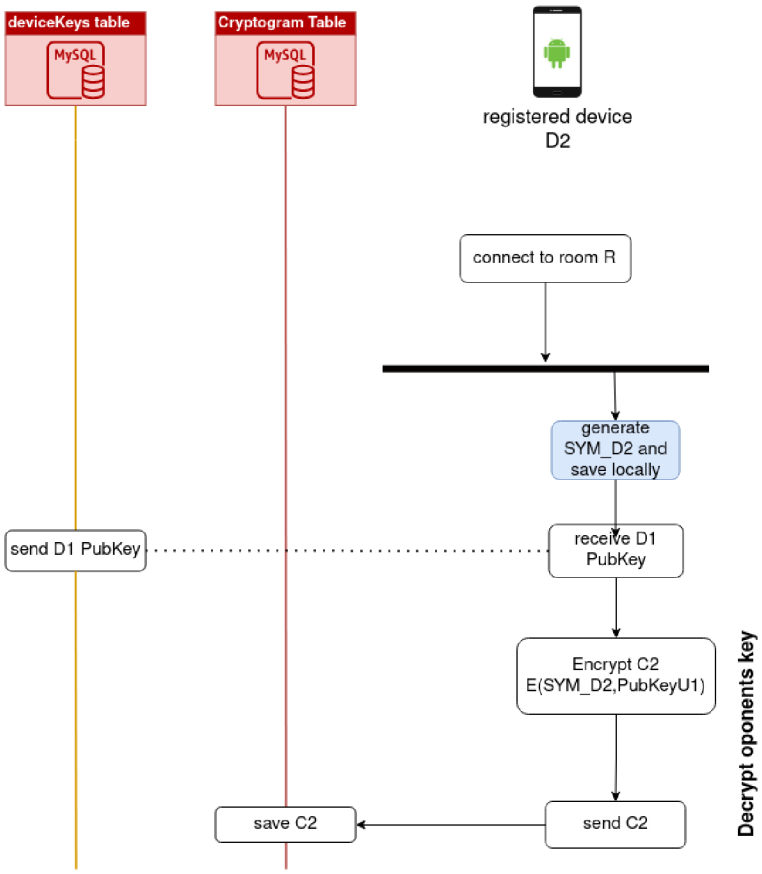
- každý užívateľ má svoj unikátny kľúč, ktorý je nutné dopraviť do zariadenia adresáta,
- každé zariadenie užívateľa má vlastné užívateľské kľúče,
- každá chatová miestnosť na zariadení má aspoň jeden symetrický kľúč, ktorým sa šifruje každá odchádzajúca správa (užívateľ má možnosť, kedykoľvek vygenerovať nový kľúč alebo zneplatniť starý).

Vytvorenie kryptogramu a odoslanie

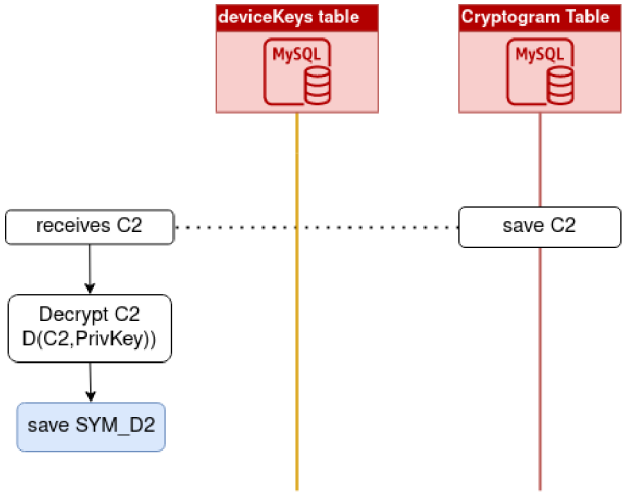
Na vygenerovanie kryptogramu je nutné mať vygenerovaný symetrický kľúč a verejný kľúč protistrany, ktorou sa symetrický kľúč šifruje. Z toho vyplýva, že prvý kto môže kryptogram vytvoriť, je užívateľ, ktorý sa do miestnosti pripája. A to z dôvodu, že užívateľ, ktorý sa pripája do miestnostim si vie vyhľadať identifikátory užívateľov s ktorými miestnosť zdieľa. Na základe identifikátorov je možné si vyžiadať verejné kľúče zariadení všetkých užívateľov v miesnosti. Po vygenerovaní symetrického kľúča je tento kľúč zašifrovaný každým verejným kľúčom protistrany a odoslané na server, ktorý to uloží to tabuľky **CRYPTOGRAMS**. Opisovaný proces je k nahliadnutiu na Obr. 4.11.

Dešifrovanie kryptogramu

Po úspešnom uložení kryptogramov zariadenia (D2) do databáze, sú o existencii daného symetrického kľúča SYM_D2 informované všetky zariadenia v miestnosti, ktorých verejným kľúčom bol SYM_D2 šifrovaný. Po obrdžaní správneho kryptogramu si ho zariadenie dešifruje pomocou SK a výsledný kľúč si uloží. Tento postup je naznačený na Obr 4.12.



Obr. 4.11: Pripojenie k miestnosti a odoslanie kryptogramu na server



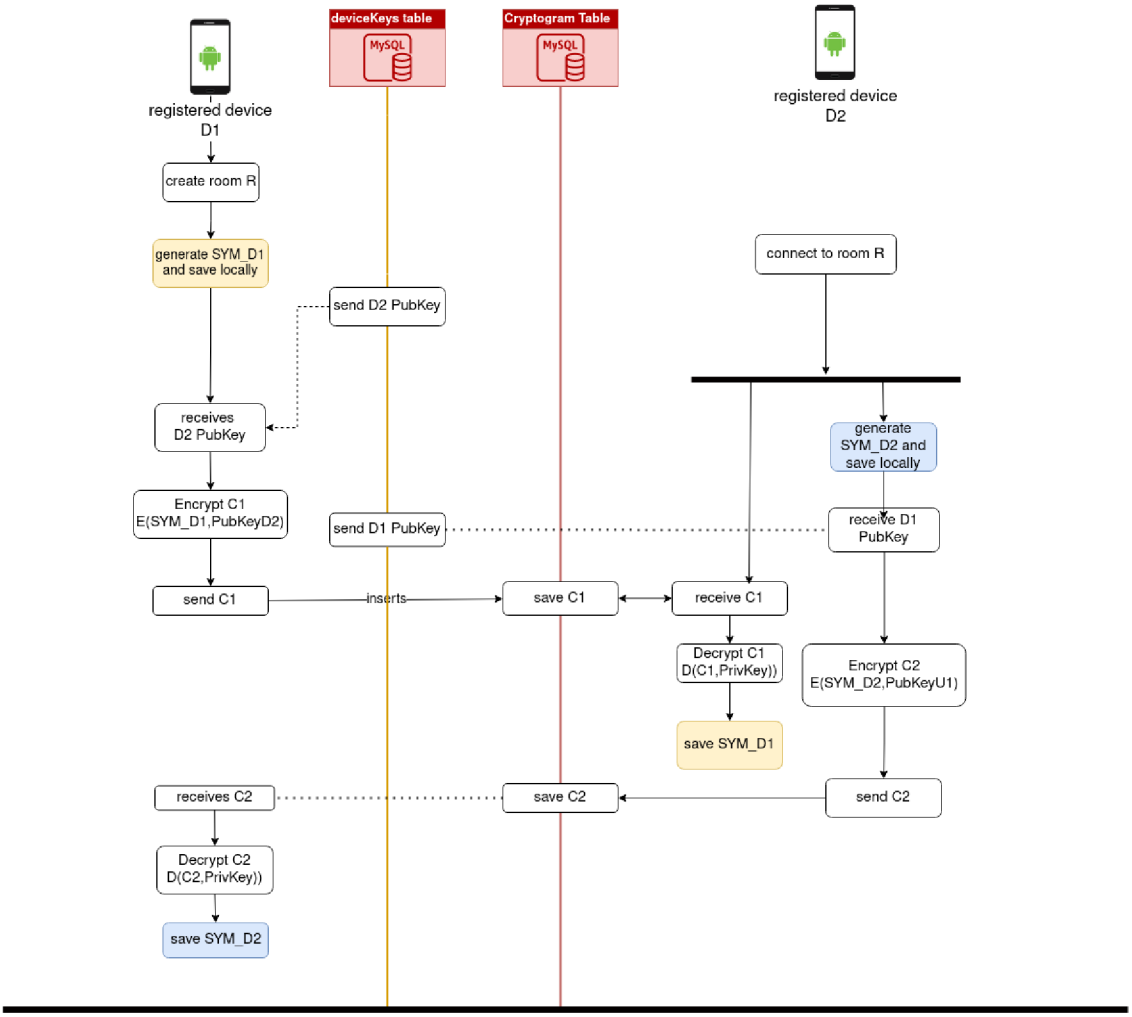
Obr. 4.12: Dešifrovanie kryptogramu

Problémy a riešenie distribúcie kľúčov

Na úspešnú komunikáciu zariadení v miestnosti, je nutné aby každý vlastnil aktuálny symetrický kľúč každého zariadenia v miestnosti. Z dôvodu zvýšeného pohodlia pre

užívateľov, bolo nutné vyriešiť problém výmeny kľúčov v časoch, kedy nie sú online všetci užívatelia v miestnosti. Práve z toho dôvodu, existuje tabuľka *cryptogram*, kde sú uložené všetky symetrické kľúče zašifrované všetkými verejnými kľúčmi zariadení v danej miestnosti.

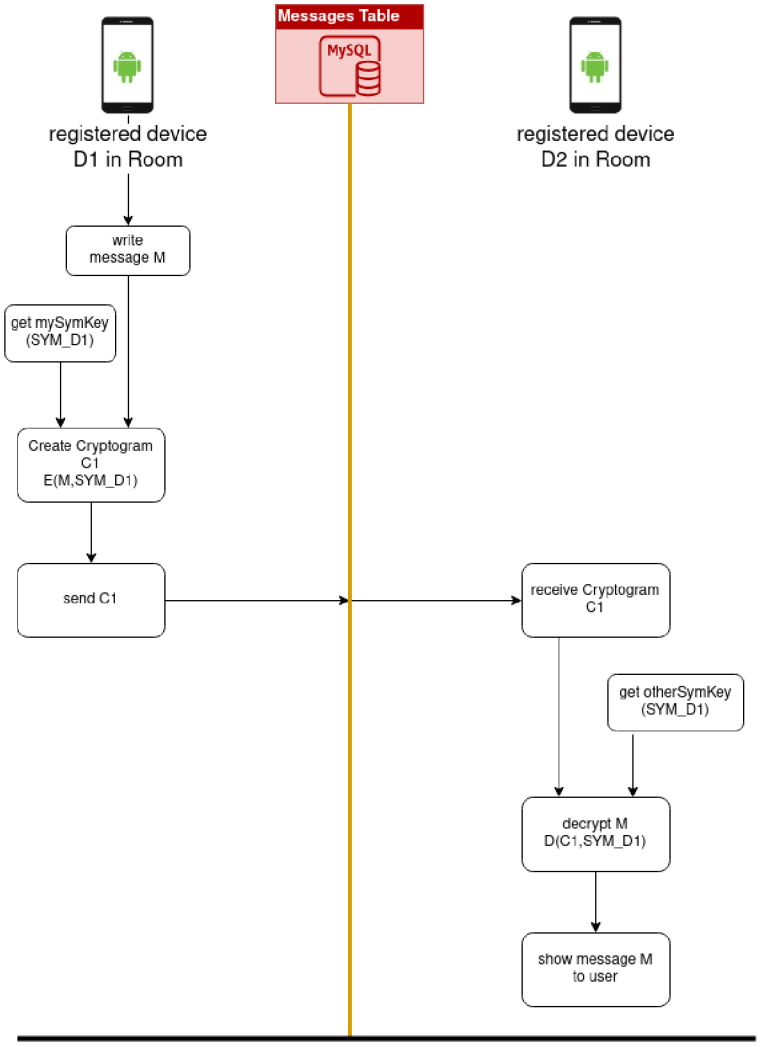
Z toho vyplýva, že užívateľ, ktorý vytvorí miestnosť, nemôže odoslať na server kryptorgam z dôvodu, že nemá ho čím zašifrovať, nemá dosah na žiaden verejný kľúč. Preto, len čaká, kým sa pripojí nový účastník, ktorý poskytne verejný kľúč na úspešné zašifrovanie kľúča a odoslanie kryptogramu. Zložitejšia logika je na strane zariadenia, ktoré sa pripája do miestnosti, totiž ono musí zašifrovať kľúč pomocou VK oponenta a zároveň dešifrovať svojím SK. Keď spojíme horeuvedené obrázky do jedného a uskutočníme proces šifrovania a dešifrovania na oboch stranách, tak výsledný návrh vyzerá ako je na obrázku 4.13



Obr. 4.13: Celé flow distribúcie kľúčov

Odoslanie správy

Odosielanie správ je jednoduchý mechanizmus, keďže sa na šifrovanie používa symetrická šifra, tak užívateľ len napíše správu, po kliknutí na *odoslať* je z lokálnej databázy prečítaný symetrický kľúč *(SYM_D1)* k danej miestnosti. Správa bude zašifrovaná symetrickým kľúčom SYM_D1, tým sa vytvorí kryptogram *C1*. Kryptogram C1, je odoslaný na server, kde je uložený do tabuľky **MESSAGES** odkiaľ si ho adresát vie stiahnuť. Po stiahnutí C1 si zariadenie nájde v tabuľke symetrický kľúč *SYM_D1* zariadenia D1, pomocou, ktorého správu dešifruje a zobrazí užívateľovi. Ukážka takejto komunikácie je na obrázku 4.14.

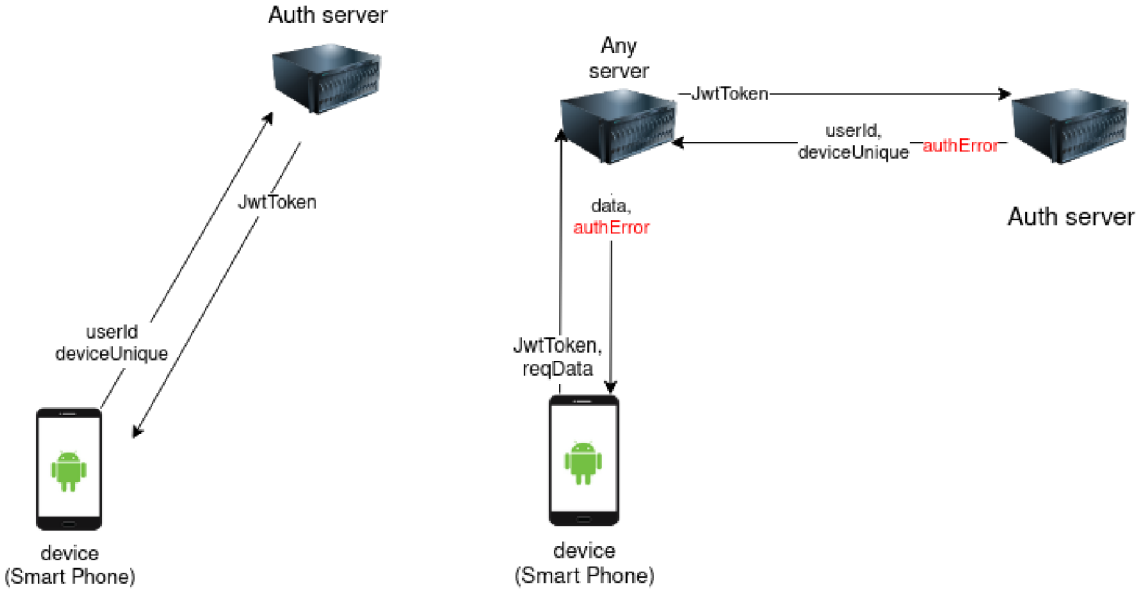


Obr. 4.14: Odosielanie správy

4.3.5 Zabezpečenie

Všetky už spomenuté komunikácie medzi zariadením (klientom) a serverom sú prenášané TCP spojením, ktoré je zabezpečené pomocou protokolu TLS. Komunikácia medzi klientom a serverom prebieha cez anonymnú sieť *TOR* 2.2. Zabezpečenie proti odchytu a zneužitiu citlivých dát sa tieto citlivé dáta prenášajú pomocou *JWT* 2.3. V JWT sú prenášané dôležité dáta ako `userID` a `deviceUnique` a časová značka platnosti tokenu. Zariadenie si pravidelne generuje token, ako je zobrazené na obrázku 4.15a, tak je odoslaný request s `useId` a `deviceUnique` na auth server, ktorý vráti platný `JWT token`, ktorým sa zariadenie bude autorizovať v komunikácii so serverom.

Po úspešnom uložení jwt tokenu do telefónu môže začať ostatná komunikácia so serverom. JWT token sa odosiela v kominikácii s každým serverom okrem auth a login serveru. Tieto dva serveri nepoužívajú žiaden z identifikátorov, ktoré token obsahuje. Komunikácia s ostatnými servermi je naznačená na Obr. 4.15b, server najskôr spracuje jwt token tým, že ho odošle na validáciu na auth server. Auth server spracuje token, skontroluje či je správne podpísaný a platnosť, ak je všetko v poriadku vráti dáta obsiahnuté v tokene v opačnom prípade vráti *authError*, ktorý sa spropaguje cez pôvodne dotazovaný server až do telefónu. Telefón na *authError* zareaguje tým, že opakuje postup generovania tokenu z Obr. 4.15a. Ak auth server žiaden error nevráti, tak pôvodne dotazovaný server obdrží dáta z tokenu (`userId` a `deviceUnique`) a pracuje s nimi dalej, po vykonaní požiadavku pošle do telefónu výsledok.



(a) generovanie tokenu

(b) overenie tokenu

Obr. 4.15: Komunikácia s auth serverom

Zabezpečenie na úrovni užívateľa telefónu bude použitý odtlačok prstu alebo bezpečná verzia štvorciferného PIN kódu. Bezpečná verzia štvortiferného PIN sa používa aj na zabezpečenie hardwarových kryptopeňaženiek. Jedná sa o štandardnú číselnú klávesnicu avšak s neštandardným náhodným rozložením znakov na klávesnici. Dokáže to zabrániť tomu, že ak útočník odsleduje akú sekvenciu obeť kliká na displej, tak útočník nedokáže z pozície na displeji vydedukovať zvolenú cifru. Možnosť odtlačku prstu, klávesnice a bezpečnejšej klávesnice bude voliteľná v nastavení aplikácie.

Ak užívateľ používa rootnuté zariadenie, tak je pomerne jednoduché dostať sa k lokálnym databázam aplikácii. V lokálnych databázach sú ukladané symetrické kľúče zariadenia a aj ostatných zariadení, čo je veľmi citlivá záležitosť, keďže s týmito kľúčami je možné dešifrovať všetky odchádzajúce a prichádzajúce správy v zariadení, preto je nutné tieto údaje chrániť. Možnosti zabezpečenia lokálnych databáz v aplikácii

1. Neukladať kľúče do DB, ale uchovávať v RAM

   -> bezpečné riešenie, ktoré predĺži načítanie správ po rozkliknutí miestnosti z dôvodu sťahovania si aktuálnych kľúčov
2. Namiesto ukladania do DB použiť možnosť serializácie (uloženie obsahu tabulky do textového súboru), Kotlin má nevýhodu, že jeho serializácia sa podobá čitateľnosťou na JSON, ale je možné využiť to, že Android podporuje jazyk C++, ktorý má možnosť serializovať aj do binárneho súboru. Binárny súbor nie je bez úprav čiteteľný pre človeka.

   -> jedná sa o menej bezpečné riešenie, lebo po dlhšom spracovaní binárneho súboru, je možné dostať sa k formátu, ktorý je čitateľný pre človeka. Jedná sa o citeľne rýchlejšiu variantu riešenia oproti vyššie spomínanej, ale za cenu toho, že útočníkovi je možné len sťažiť cestu ku kľúčom, nie úplne eliminovať hrozbu
3. Do DB ukladať symetricky šifrované dáta kľúčom, ktorý bude zaserializovaný v jazyku C++ ako bolo spomínané v bode *2*.

   -> Jedná sa o rýchlejšiu variantu než je spomínaná v bode 2
4. **Android ponuka riešenie, ktorým je šifrovanie databáz pomocou androidovskej knižnice, jedná sa asi o najviac optimálnu cestu riešenia, avšak taktiež neposkytuje istotu nedešifrovania databáz z dôvodu, že heslo k databázam bude musieť byť uložené v zariadení.**

Riešením by bola logická štruktúra ako tieto heslá udržiavať na servery a zariadenie by sa k nim pri štarte aplikácie dotazovalo. Bezpečnosť by bolo možné umocniť tým, že by sa toto heslo v pravidelných inervaloch menilo. Prípadne ak by bola požiadavka neukladať kľúč od DB na server, bolo by možné využiť užívateľský PIN do aplikácie, prípadne vždy pri spustení aplikácie by bolo od užívateľa vyžadované heslo, ktoré si určil na šifrovanie databáz. Ale ak by bola táto réžia nechaná na užívateľa, vzniká tu slabý článok v systéme, či už z hľadiska stratenia dát alebo nedôsledného stráženia hesla užívateľom. Po strate hesla k DB, by bolo možné aplikáciu používať až po opätovnom prihlásení do systému po úspešnej autorizácii (muselo by prebehnúť odhlásenie zo zariadenia).

V návrhu aplikácie je, že užívateľ si bude môcť vybrať, ktorý zo spôsobov zabezpečenia kľúčov bude preferovať. V ponuke bude varianta *1, 3 a všetky podvarianty z bodu 4* .

Zabezpečenie na úrovni dôvery v server, je taká, že by užívateľ pri prvom spustení aplikácie určiť, ku ktorému serveru sa bude zariadenie pripájať, de facto na ktorom serveri budú uložené užívateľove kľúče. Čo môže znamenať, že niekto, kto si stiahne nakonfiguruje a spustí servre vytvorené v tejto práci, si po prihlásení do aplikácie vpíše cestu k svojim servrom a všetka komunikácia, ktorá prebehne bude ukladaná tam. Všetko okrem dát užívateľa. Problém je ten, že všetci s kým bude chcieť komunikovať, sa musia dotazovať na rovnaký server.

4.3.6 Notifikácie

Pre užívateľsky prívetivý chod aplikácie je nutné implementovať notifikácie. Užívateľ obdrží notifikáciu vždy ak nastane dôležitá udalosť, ktorá sa ho týka. Notifikácia je posielaná ak nastane akcia

- pripojenie nového člena do miestnosti
- nová správa v miestnosti

Na úspešné doručenie notifikácie na zariadenie je potrebné mať uložený notifikačný token - firebaseToken. Tento token je generovaný knižnicou, ktorá je implementovaná v aplikácii. Po úspešnom prihlásení užívateľa nastáva so všetkým čo je opisované v kapitole 4.3.2 aj odosielanie firebaseTokenu. Z toho vyplýva, že v tabulke **DEVICEKEYS** je okrem `userId`, `deviceUnique`, `VK` aj `firebaseToken`. Firebase token je unikátny identifikátor zariadenia v celej sieti zariadení, jeho vygenerovanie sa nedá vynútiť, vygeneruje sa nový vždy keď si to vyžiada knižnica. Vždy, keď je vygenerovaný nový token tak je prepísaný token v tabulke **DEVICEKEYS** pre dané zariadenie.

5 Backend aplikácie

Každá aplikácia komunikujúca s internetom potrebuje mať prostredníka, svoj server. Server tejto aplikácie beží na Google cloud. Beží na nej viacero nezávislých blokov - dockerov 2.5, každý má svoju špecifickú úlohu. Pre funkčnosť aplikácie je potrebné mať niekde bežiacu inštanciu nasledovných docker imageov

- Golang - pre podporu jazyka Golang
- Postgres - jedná sa o databázu PostgreSQL
- Python - pre podporu jazyja Python
- **grpcUsers** - má na starosti prihlásenie a registráciu užívateľov
- **grpcAuth** - generuje a overuje JWT tokeny 2.3
- **grpcDeviceTokens** - spracúva réžiu verejných kľúčov a kryptogramov symetrických kľúčov
- **grpcRooms** - má na starosti réžiu miestností
- **grpcMessages** - obsluhuje správy užívateľov
- **pyNotificationServer** - slúži na posielanie notifikácií na zariadenia

Docker image, ktoré nie sú v zozname zvýraznené hrubo, sú len stiahnuté, spustené a nakonfigurované, všetky ostatné sú vytvorené čisto len pre účel tejto práce.

5.1 User server

Jedná sa o docker image, ktorý obsluhuje registráciu a prihlásenie. Je to priama implementácia logiky, ktorá je vysvetľovaná v kapitole 4.3.1. Po autorizácii užívateľa a po úspešnom zistení užívatelského id - `userId` je tento server nepotrebný až do okamyhu, keď sa užívateľ odhlási a bude sa chcieť opätovne prihlásiť.

Môžu nastať nasledovné chyby:

- **USER_NOT_FOUND - 404** - user not found nastane, keď užívateľ vloží kombináciu, ktorá nepatrí žiadnemu užívateľovi
- **USER_ALREADY_EXIST - 409** - user already exist nastane ak sa registruje užívateľ, ktorý je už registrovaný

Oba druhy chýb sú bez opakovania requestu zobrazované užívateľovi.

5.2 Auth server

Ďalší server, s ktorým užívateľ príde do kontatu je auth server. Tento docker image je jeden z najvyťaženejších a to z dôvodu, že každé zariadenie s ním vedie inenzívnu komunikáciu, buď na priamo alebo sprostredkovane z iného serveru.

Možnosti komunikácie s auth serverom:

- **na priamo** - slúži len na vygenerovanie JWT tokenu ako je opisované v kapitole 4.15a.
- **sprostredkovane** - jedná sa o najčastešiu komunikáciu s týmto serverom, kde väčšina komunikácie medzi zariadením a servermi je doprevádzaná JWT tokenom, ktorý je treba si overiť, takže tento scenár je implementovaný presne tak, ako je naznačené na Obr. 4.15b.

Slúži len na vygenerovanie tokenu s dátami a na overenie a vytiahnutie dát z tokenu. Návratová hodnota pri overovaní tokenu môže byť:

- *UNEXPECTED_TOKEN* ***406*** - unexpected token signing method, jedná sa vadný token, ktorý zrejme nie je súčasťou systému, lebo šifrovanie v tokene nesúhlasí so spôsobom šifrovania na serveri. Na šifrovanie sa používa podpis typu HMAC-SHA-256.
- *UNAUTORIZED_TOKEN* ***401*** - Invalid token, jedná sa o token so správnym typom šifrovania, avšak po overení klúčom, ktorým sú šifrované tokeny v systéme sa ukázal ako neplatný alebo už uplynula doba platosti
- *UNACCEPTABLE_TOKEN* ***401*** - Invalid token claims, jedná sa o chybu, ktorá tvrdí, že sú neplatné alebo poškodené dáta v tokene (userId a deviceUnique).

Chyby sa spropagujú až do zariadenia užívateľa, návratové chyby s kódom **401** sa vygeneruje nový token, s ktorým sa požiadavka zopakuje. Ak by sa náhodou nepodarilo vygenerovať token lebo by sa chybová hláška tri krát opakovala s rôznymi tokenmi, tak aplikácia zobrazí systémovú chybu. V prípade návratu chyby s kódom **406**, tak sa rovno zobrazuje systémová chyba a to z dôvodu, že zrejme sa zmenila technológia podpisu tokenu a od užívateľa bude vyžadovaná bud aktualizácia aplikácie alebo prinajhoršom reštart aplikácie.

5.3 Device token server

Ďalším, zrejme najdôležitejším docker imageom v systéme je deviceTokenServer, jedná sa o server spravujúci tabuľky obsahujúce kľúče a kryptogramy kľúčov. Zastrešuje logiku opisovanú v kapitole 4.3.2 a plní tabuľku **DEVICEKEYS** užívateľskými údajmi

- `userId` - unikátny identifikátor užívateľa, ktorého funkcia je vysvetlená v kapitole 4.3.2.
- 4.3.2 - jedná sa o verejný kľúč asymetrickej kryptografie, konkrétne RSA
- `device_unique` - unikátny identifikátor zariadenia, ktorého funkcia je opisovaná taktiež v 4.3.2.
- `firebase_token` - funkčnosť tohto tokenu je detailnejšie opisovaná neskôr v texte, konkrétne v kapitole 5.6

Funkcie obsluhujúce túto databázu dokážu vrátiť chyby z auth serveru (opisované v kapitole 5.2) a tieto chyby prepošle zariadeniu.

Ďalšia tabuľka, ktorá je plnená touto službou je *cryptograms*, ktorej logika je detalne opísaná v 4.3.4. Ohladom chybových stavov, ktoré môžu nastať pri obsluhovaní spomínanej tabuľky sú to chybové kódy, ktoré vráti authServer obohatené o:

- **QUERY_ERROR - 501** - jedná sa o chybu, ktorá je vrátená zariadeniu v prípade, že nebol úspešne spustený dotaz na vyhladávanie v tabuľke a to zle naísaným dotazom do databáze
- **UNKNOWN_ERROR - 502** - jedná sa o najgenericejšia chyba, ktorá indikuje neošetrenú chybu

Chyby ktoré majú kódy, ktoré sú väčšie než 500 sú chyby za ktoré je zodpovedný server, takže aplikácie reaguje zobrazením systémovej chyby. Tento typ chýb by sa nemal vyskytovať v systéme, avšak je nutné ošetriť aj predpoklad úplného pokazenia systémov spôsobených zlou konfiguráciou alebo právami.

Tento server využíva výhod tecnologie GRPC 2.7.2, ktorá poskytuje možnosť vytvoreného stremu. Stream je tok dát, ktorý je priamo medzi klientom a serverom, pričom počas spojenia je možné kedykoľvek odoslať dáta, ktoré klient prijme. Takže na jednu žiadosť od klienta je možné poslať N odpovedí.

Tento stream sa používa na informovanie užívateľov o tom, že oponent (v chatovej miestnosti) zverejnil nový kryptogram alebo verejný kľúč, v takých prípadoch je na zariadenie odoslaná informácia o tom, že nastala zmena a zariadenie si zmeny stiahne a pracuje s nimi. Na základe tejto technológie je umožnené čakať vo vytvorenej miestnosti na akciu pripojenia oponenta a akonáhle sa oponent do miestnosti pripojí tak bez potreby obnovenia stránky sa stránka obnoví sama.

5.4 Rooms server

Tento server obsluhuje všetok tok okolo miestností

- **Vytvorenie miestnosti** - užívateľ vytvorí miestnosť a tvorcu do nej automaticky vloží do databáz ako je zobrazené na Obr. 4.9
- **Pripojenie k miestnosti** - užívateľ vyplní kód, ktorý mu poslal iným médoim kamarát a na základe neho sa pridá do miestnosti v ktorej sa kamarát už nachádza. Toto je zobrazené na Obr. 4.10

Obe tieto funkcie miestností je detailne opísané v 4.3.3.

Server okrem authErrorov 5.2 server používa

- **TEMP_CONNECTOR_NOT_FOUND_MATCH - 407** - Room not found, chyba ktorú dostane zariadenie, ktoré sa chce pripojiť s tempConnectorom, ktorý nebol v systéme nájdený
- **WRONG_CURSOR - 403** - Empty cursor, chyba, ktorá oznámi zariadeniu že poslal zlý parameter do funkcie, ktorá listuje miestnosti užívateľa
- **ERROR_WHILE_CONVERT_TO_INT64 - 502** - chyba ktorá nastane, keď sa ako parameter pošle sekvencia znakov, ktorá nejde pretransformovať na číslo
- **ERROR_WRONG_TIMSTAMP - 403** - nastáva keď je do funkcie listovania všetkých miestností užívateľa zlý formát času
- **UNKNOWN_ERROR - 505** - jedná sa o najgenericejší error, ktorý indikuje neošetrenú chybu

Znovu platí, že chyby ktoré majú kódy, ktoré sú väčšie než 500 sú chyby za ktoré je zodpovedný server, takže aplikácie reaguje zobrazením systémovej chyby. Tento typ chýb by sa nemal vyskytovať v systéme, avšak je nutné ošetriť aj predpoklad úplného pokazenia systémov spôsobených zlou konfiguráciou alebo právami.

Tento server taktiež implementuje funkciu streamu, ktorú umožňuje technológia GRPC. Konkrétne použitie streamov je na tvorbu a párovanie miestností. Užívateľ, ktorý vytvorí miestnosť a odošle kód kamarátovi, tak ostáva na obrazovke kde je kolekcia miestností a akonáhle sa kamarát do miestnosti pripojí, tak server pomocou streamu pošle do zariadenia informáciu o tom, že nastala zmena, na ktorú reaguje zariadenie znovu stiahnutím užívateľových miestností. Po opätovnom stiahnutí tvorca miestnosti vidí, že už je v nej niekto pripojený. Rovnakým spôsobom funguje aj pripájanie sa do miestnosti. Po úspešnom pripojení server po čase pripájanému zariadeniu pošle informáciu o tom, že nastala zmena, nech sa zaktualizuje.

Na réžiu celej aplikácie nestačí mať na tomto serveri len opisované funkcie ale aj množstvo doprovodných funkcií, ktoré zisťujú užívateľov v miestnosti na základe roomId a rôzne variácie tejto funkcie (všetkých okrem mňa, všetky moje zariadenia...). Tieto funkcie sú potrebné pre funkciu ostatných serverov.

5.5 Message server

Zrejme najdôležitejším serverom je server, ktorý má na starosti správy, plní tabuľku **MESSAGES**. V tabuľke messages sa nachádzajú kryptogramy jednotlivých užívateľských správ spolu s miestnosťou, do ktorej patria a identifikátorom užívateľa, ktorý správu odoslal a deviceUnique zariadenia z, ktorého bola správa odoslaná. Tok samotných správ nie je vôbec náročný, pomocou GRPC streamov si zariadenie

vytvorí spojenie so serverom, ktorý ho vďaka poskytnutým parametrom subscribne na správy v miestnosti a vždy keď sa do miestnosti odošle nová správa, tak stream o tom upozorní zariadenie, ktoré na to patrične zareaguje.

Jedná sa o jednoduchú službu, ktorá len reflektuje dianie sa na serveri a posiela informácie o zmene prípadne posiela dáta pre kolekciu správ. Pracuje len s kryptogramami, ktoré sa dešifrujú priamo na zariadení pomocou dát, ktoré mu poskytnú ostatné servery.

5.6 Notification server

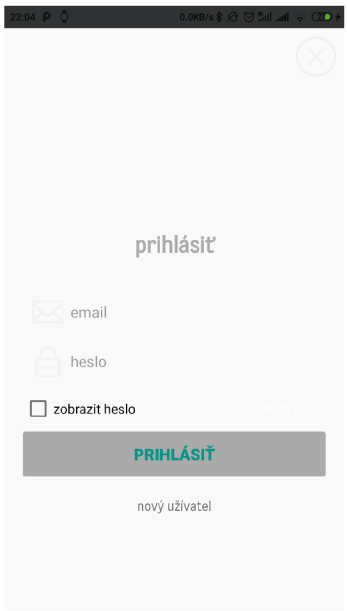
Existujú dva druhy notifikácií, ak je aplikácia spustená alebo ak je aplikácia vypnutá. Notifikácie ktoré sa zobrazujú v aplikácii počas spustenej aplikácie sú riešené pomocou GRPC streamov viz Obr. 2.1. Pre druhú variantu je nutné použiť notifikačný server. V momente ak je nutné odoslať notifkáciu na zariadenie, napríklad z dôvodu novej správy sa do notifikačného serveru pošle request, ktorý obsahuje typ notifikácie, všetky firebase tokeny užívateľov, ktorým má notifikácia doraziť a text notifikácie, ktorý užívateľ uvidí v notifikačnej lište telefónu. Na základe týchto údajov server odošle request do platformy Firebase 2.4, ktorá zabezpečí rozoslanie notifikácii na zariadenia s požadovaným firebaseTokenom.

6 Výsledná aplikácia

V tejto kapitole je detailnejšie opísaná aplikácia a všetky jej obrazovky.

6.1 Autentizácia

Po nainštalovaní aplikácie je nutné sa do nej prihlásiť, buď formou vytvorenia účtu alebo prihlásenia do existujúceho. Na prihlásenie slúži obrazovka Obr. 6.1, kde užívateľ vyplní svoje meno a heslo, ktoré sú následne odoslané na server Obr. 5.2, ktorý buď vráti userId a užívateľské nastavenia alebo jednu z už spomínaných chýb.



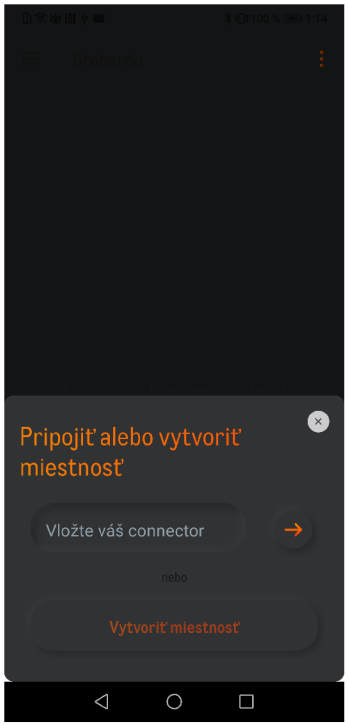
Obr. 6.1: Prihlásenie do aplikácie

6.2 Miestnosti

Po registrácii užívateľ nepatrí do žiadnej miestnosti, preto je nutné ju vytvoriť. Užívateľ vytvorí miestnosť pomocou tlačidla v pravom dolnom rohu obrazovky. Po kliknutí na tlačidlo sa otvorí dialóg, ktorý je vyobrazený na Obr. 6.2. Na dialógu si užívateľ môže vybrať z dvoch možností

- **Vytvoriť miestnosť** - užívateľ odošle na server request na vytvorenie miestnosti, táto akcia bola opisovaná v kapitole 4.3.3 a 5.4.
- **Pripojiť k miestnosti** - Po vložení *tempConnectoru* je možné požiadať o pripojenie do danej miestnosti

Toto po odoslaní žiadosti o pripojenie je možné, že server vráti jednu z chýb 5.4, tak je chyba oznámená užívateľovi pomocou chybovej hlášky v snackbare. V prípade, že nastane chyba, ktorá nie je vrátená serverom, ale jedná sa o chybu s internetovým pripojením, tak sa zariadenie pokúsi tri krát požiadavok zopakovať a ak je aj na tretí pokus neúspešný, tak sa užívateľovi zobrazí hláška o žiadnom alebo slabom internetovom pripojení. V prípade komunikácie so serverom, ktorá neskončí chybou, sa zaktualizuje kolekcia konverzácii užívateľa a kolekcia bude obsahovať novú/novo spojenú miestnosť.



Obr. 6.2: Dialóg miestostí

6.3 Inicializácia konverzácie a komunikácia

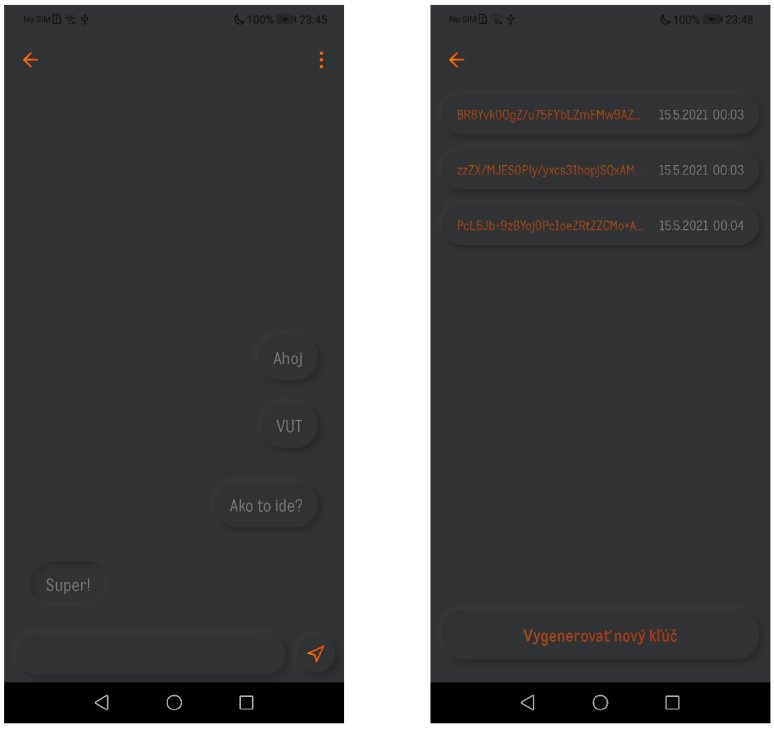
Po úspešnom pripojení do miestnosti sa vykonáva všetka logika výmeny kľúčov, ktorá bola vysvetľovaná v kapitole 4.3.4. Z hľadiska vysvetlenia a funkčnosti je tam množstvo stavov a možností, avšak pre užívateľa aplikácie je jednoduché odoslať správu, lebo stačí len otvoriť miestnosť, do ktorej má byť správa odoslaná, napísať text správy do správneho políčka a kliknúť na tlačidlo odoslať. O všetko ostatné sa postará flow aplikácie.

Adresátovi správy je zobrazená notifikácia o tom, že v miestnosti je nová správa. Po kliknutí na notifikáciu alebo manuálnom otvorení miestnosti s novou správou. Po

otvorení miestnosti sa najskôr stiahnu kryptogramy, ktoré sú zobrazené užívateľovi prakticky hneď, následne sa každý z nich začne dešifrovať a miesto kryptogramov sa po čase zobrazí dešifrovaná správa. V prípade, že kľúč k danému kryptogramu bol zmazaný alebo sa mu skončila platnosť, tak adresátovi bude vyobrazený len kryptogram, čo znamená, že uvidí len sekvenciu znakov, ktoré nedávajú žiadnu výpovednú hodnotu.

Každý užívateľ si môže z každého zariadenia sám režírovať platnosť svojich kľúčov. V detaile miestnosti je napravo hore možnsoť otvorenia menu Obr. 6.3a, v ktorom sa zatiaľ nachádza len možnosť réžie kľúčov Obr. 6.3b. Na obrazovke réžie kľúčov si užvateľ môže spravovať kľúče svojho aktuálneho zariadenia. Je mu umožnené:

- **zneplatniť** jeho symetrický kľúč (zariadenie musí byť online), následne vďaka réžii kľúčov sa táto informácia rozdistribuuje do všetkých zriadení a správy odoslané z tohto zaradenia nebude možné dešifrovať,
- **predĺžiť platnosť** kľúču, kľúč, ktorého platnosť expirovala, je automticky zmazaný zo systému a zariadenie si musí vygenerovať nový,
- **vygenerovať nový** kľúč, ktorým budú šifrované všetky od toho momentu odoslané správy (starý kľúč je stále platný, pokiaľ nebude vymazaný alebo neexpiruje).



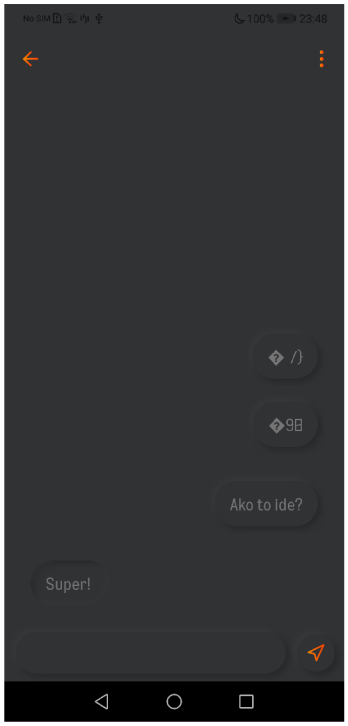
(a) Správy v miestnosti (b) Réžia kľúčov

Obr. 6.3: Možnosti v miestnosti

6.3.1 Mazanie kľúčov

Na Obr. 6.3a je zobrazená komunikácia medzi dvoma užívateľmi. Pre jednoduchosť uvažujme jedno zariadenie u každého užívateľa, každé zariadenie má svoj symetrický kľúč, ktorým šifruje každú odosielanú správu. Správy "ahoj" a "VUT" sú šifrované prvým klúčom v kolekcii viz Obr. 6.3b. Po odoslaní týchto dvoch správ užívateľ manuálne (klikom na tlačidlo) dal vygenerovať nový šifrovací kľúč, ktorým bola šifrovaná správa "Ako to ide?". Všetky správy sa šifrované dostali k oponentovi ktorý prispel do konverzácii tiež. Následne užívateľ zmazal prvý kľúč z kolekcie, čím odstránil jediný klúč k daným správam a všetky správy šifrované daným kľúčom sa už nikdy neobrazia v dešifrovanej podobe, len ako kryptogram ako je zobrazené na Obr. 6.4 s pohľadu odosielateľa. Po odstránení kľúča sa zmaže kryptogram z tabuľky, tým pádom, každé zariadenie, ktoré si od toho momentu zažiada o zaslanie kryptogramov symetrických kľúčov neobdrží už zmazaný kľúč. Ak zariadenie daný kľúč už má, tak to znamená, že aktuálne má otvorené správy v chatovej miestnosti, v tom prípade server odošle na zariadenie informáciu o tom, že nastala zmena v dešifrovacích kľúčoch. Na túto informáciu aplikácia reaguje tým, že sa stiahnu všetky aktuálne klúče v miestnosti a všetky zmazané alebo v tom momente už neplatné kľúče sa odstránia z DB v zariadení. (pozn. odstránenie z DB platí len vo variante, ak sa na danom zariadení využíva varianta s uchovávaním klúčov v zariadení viz Obr. 4.3.5)

Vizualizácia správ respektíve kryptogramov správ, ktoré sú uložené v DB na serveri zobrazené na Obr. 6.5.



Obr. 6.4: Správy v miestnosti po odstránení dešifrovacieho kľúču

21:39:01.1…	2021-05-14 21:39:01.1…	…	74	4	pYI4Hg==	s	…	d5ef764a9f63dc7c	3	0
21:39:10.2…	2021-05-14 21:39:10.2…	…	74	4	sr8D	s	…	d5ef764a9f63dc7c	3	0
21:42:06.7…	2021-05-14 21:42:06.7…	…	74	3	BZXBZ1AWFg==	s	…	987cc1b9e7888b1c	3	0
21:40:55.2…	2021-05-14 21:40:55.2…	…	74	4	XTjla07sWRWs/8M=	s	…	d5ef764a9f63dc7c	3	1

Obr. 6.5: Kryptogramy správ

7 Zhrnutie

Výsledná aplikácia implementuje množstvo funkcií, ktoré užívateľovi zabezpečujú bezpečnosť v sieti, istotu v tom, že správu, ktorú odošle prečítajú len adresáti. Aplikácia sa dá považovať za priamu konkurenciu množstvu sociálnych sietí, ktoré na bezpečnosť užívateľov neprihliadajú, dokonca ťaží z údajov, ktoré o svojich užívateľoch má.

Porovnanie aplikácii radiacich sa pod typ instant mesasging a výslednej aplikácie tejto práce z hľadiska užívateľských funkcií. Tabuľka 7.1, kde sú popisované jednotlivé funkcie aplikácii. Pri aplikácií Grobardo (aplikácia, ktorá je výsledkom tejto práce) je funkcia opísaná slovom "čoskoro"znamená, to že sytém je priprvený na implementáciu danej funkcionality, avšak zatiaľ sa v systéme nevyskytuje.

Tab. 7.1: Porovnanie funkcionalít jednotlivých IM aplikácii

	Whatsapp	Threema	Facebook	The Signal	Telegram	Grobardo
chat	✓	✓	✓	✓	✓	✓
skupinový chat	✓	✓	✓	✓	✓	✓
telefonat	✓	✓	✓	✓	✗	čoskoro
videohovor	✓	✓	✓	✓	✓	✗
integracia kontaktov z FB	✗	✓	✓	✗	✗	✗
úprava vzhladu	✓	✗	✓	✓	✓	✗
zdielanie súborov	✓	✓	✓	✓	✓	čoskoro
nutnosť tel.č	✓	✗	✗	✓	✗	✗

Užívateľské funkcie sú užitočné v aplikácii avšak oveľa dôležitejšie je to čo je na ich pozadí. Bezpečnosť sa odohráva práve na pozadí užívateľských funkcií. V tabuľke 7.2 sú porovnané najčastejšie používané instant messaging aplikácie v porovnaní s aplikáciou je zobrazené porovnanie aplikácie tvorenej v tejto práci. Je nutné okomentovať pár bodov v tejto tabuľke 7.1, najmä riadok, kde je porovnávaná funkcionalita *dešifračný kľúč uložený v zariadení.*

U aplikácie *Whatsapp* je udaná hodnota áno aj nie, to z dôvodu, že kľúč v zariadení je, avšak nejedná sa o jediný kľúč, ktorým je možné správu dešifrovať. Whatsapp umožňuje stiahnuť si predošlé správy do nového zariadenia aj napriek tomu,

že ak by boli zašifrovan E2E, tak by správy nemali byť čitateľné. To, že tieto správy čitateľné sú, je dôkaz toho, že jestvuje na serveri kľúč, ktorým je možné správy dešifrovať poprípade prešifrovať na správyčitateľné pre nové zariadenie. Prípadne aplikácia Telegram implementovala funkciu, ktorou je možné si do siete Telegram preniesť všetky historické správy z Whastappu. Funguje to tak, že je možné si stiahnuť všetky správy do súboru .zip, tieto správy si následne užívateľ naimportuje do Telegramu a má ich tam.

U riadku *šifrovanie metadát* je u aplikácii Grobardo udané, "nieä to z dôvodu, že jediné metadáta, ktoré sú používané v platforme sú informácie o type správy (text, obrázok, link, video, súbor).

Pod pojmom *užívateľská réžia kľúčov* je myslená moc užívateľa zneplatniť daný šifrovaný kľúč prípadne vygenerovať nový. Aplikácia Grobardo túto funkcionalitu má.

Úplné odstránenie odoslaných správ znamená možnosť užívateľa definitívne odstrániť správu z databáz, všetkých diskov. Inak povedané úplné odstránenie z internetu. Tu treba spomenúť aplikáciu The Signal, pri ktorých je udávaná hodnota "áno". Je to len z dôvodu predpokladu, že túto funkciu implementuje, nebol dohľadaný žiaden dôveryhodný zdroj, ktorý by tvrdenie vyvrátil.

Zneprístupnenie zariadení jedná sa o možnosť odstrániť prístup k aplikácii po odcudzení zariadenia alebo prípadného odhalenia hesla. Aplikácia Grobardo problematiku rieši odstránením prístupových a dešifračných kľúčov na danom zariadení. Napríklad aplikácia Messenger od Facebooku problematiku rieši odhlásením daného zariadenia z účtu. Toto riešenie má slabinu v tom, že ak je zariadenie offline, tak tento zásah nebude mať žiaden vplyv. Aplikácia si velké množstvo správ ukladá do lokálnej pamäte, takže je možné si ich prečítať offline priamo v aplikácii, prípadne po pripojení zariadenia k počítaču. Obdobne to riešia aj ostatné aplikácie.

Pod pojmom *réžia práv oponenta* je možné si predstaviť situáciu, že je odoslaná oponentovi nejaká fotka a na to, aby si ju on mohol stiahnuť alebo spraviť snímok obrazovky je nutné povolenie odosielateľa. Pre video, súbor, text a ostatné typy správ je správanie obdobné t.j nutnosť povolenia autora. Toto je jedná z funkcii, ktorá v sledovaných aplikáciach nie je. Avšak treba spomenúť, že aplikácia Snapchat podobnú funkcionalitu už implementuje.

Tab. 7.2: Porovnanie funkcionalít jednotlivých IM aplikácii

	Whatsapp	Threema	facebook	The Signal	Telegram	Grobardo
poskytovanie dát tretím stranám	✓	✗	✓	✗	✓	✗
uchovávanie a zber užívateľských dát	✓	✗	✓	✗	✓	✗
privacy by default	✗	✓	✗	✓	✓	✓
opensource	✗	✗	✗	✓	✓	✓
dešifračný kľúč uložený v zariadení	✓ (✗)	✓	✗	✓	✓	✓
čitateľnosť správ pre poskytovateľa	✓	✗	✓	✗	✗	✗
šifrovanie metadát	✗	✓	✗	✓	✓	✗?
sekundárny faktor overenia	✗	✓	✗	✓	✗	✓
logovanie času a IP	✗	✗	✓	✓	✗	✗
samozničujúce sa správy	✓	✗	✓	✓	✓	✓
užívateľská réžia kľúčov	✗	✗	✗	✓	✗	✓
úplné odstránenie odoslaných správ	✗	✗	✗	✓	✗	✓
zneprístupnenie zariadení	✓	✗	✓	✓	✗	✓
réžia práv oponenta	✗	✗	✗	✗	✗	✓

Záver

V semestrálnej práci bola vysvetlená problematika zabezpečenia údajov v Instant Messaging aplikáciach, bol vysvetlený pojem metadata a s ním spojené riziká. Prebehla detailnejšia analýza bezpečnostných rizík na strane serveru aj na strane užívateľa. Boli preskúmané aktuálne najpoužívanejšie Instant Messaging apliácie, bola porovnávaná ich funkcionalita a zabezpečenie. V systéme sú používané nasledovné kryptografické primitíva: na tvorbu hashov na strane serveru aj smartfónov je zvolená funkcia SHA-256, ako symetrická šifra, ktorou bude šifrovaná každá správa je zvolené AES-256 a na distribúciu symetrických kľúčov medzi užívateľmi je používané RSA-2024.

Výsledkom práce je návrh kryptosytému pre IM aplikáciu, ktorý je implementovaný v mobilnej aplikácii na operačný systém Android. K aplikácii bolo nutné vytvoriť backend, ten sa skladá z viacero serverov. Každý server má na starosti špecifickú časť systému, za týmto účelom vzniklo päť servrov, ktoré majú na starosti, užívateľov, miestnosti, výmenu kľúčov, generovanie a overovanie prístupových tokenov a notifikačný server. Všetky servre okrem užívateľského sú plne klonovatelné a po zmene konfigurácie aplikácie, je možné pripojiť sa na zklonovaný server, ktorý môže mať skupinka užívateľov pod vlastnou réžiou. Kľúčové funkcie systému boli overené a majoritne plne implementované do systému. Systém zvláda generovanie, šifrovanie, dešifrovanie, odosielanie, predlžovanie, mazanie a výmenu kľúčov a to symetrických aj asymetrických. Ďalej podporuje odosielanie, prijatie, šifrovanie a dešifrovanie textových správ, s týmto je spojený aj update správ v miestnosti a notifikácie na zariadenie, ktorému prišla nová správa. Dôležitou súčasťou aplikácie je vytváranie a pripájanie sa do miestností. Takže všetko čo slúži na spájanie užívateľov, výmenu správ, réžiu kľúčov a ostatné funkcie sú navrhnuté a pripravené na implementáciu v telefóne alebo len pripravené na implementáciu na serveroch.

Literatúra

[1] *Facebook Q4 and full Y2020* https://investor.fb.com/investor-news/press-release-details/2021/Facebook-Reports-Fourth-Quarter-and-Full-Year-2020-Results/default.aspx [cit. 2019-12-17]

[2] RASTOGI, Nidhi; HENDLER, James. *WhatsApp security and role of metadata in preserving privacy. arXiv Prepr. arXiv1701, 2017, 6817: 269-275* [cit. 2019-12-17]

[3] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. *Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7* [cit. 2019-12-17]

[4] Badoo webpage. *Https://badoo.com/privacy/* [online]. Internet: Badoo, 2019 [cit. 2019-12-17]. Dostupné z: https://badoo.com/privacy/ [cit. 2019-12-17]

[5] *Porovnanie sociálnych sietí* [online]. [cit. 2019-12-17]. Dostupné z: http://socialcompare.com/en/comparison/app-mesaging-31hwi0vj [cit. 2019-12-17]

[6] *Porovnanie bezpečnosti socialnych sietí* [online]. quora: Maria Dyatlova, 2014 [cit. 2019-12-17]. Dostupné z: https://www.quora.com/Is-Telegram-messenger-secure-Is-it-more-secure-than-WhatsApp [cit. 2019-12-17]

[7] *Secure Messaging Apps Comparison* [online]. [cit. 2019-12-17]. https://www.securemessagingapps.com/ [cit. 2021-5-14]

[8] *WhatsApp faq* [online]. faq.whatsapp.com: WhatsApp, 2019 [cit. 2019-12-17]. Dostupné z: https://faq.whatsapp.com/en/android/28030015/

[9] *An Anonymous, Asynchronous, Serverless Instant Messaging Protocol* [online]. faq.whatsapp.com: WhatsApp, 2019 [cit. 2019-12-17]. Dostupné z: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1625context=hicss-52

[10] Encryption on Facebook Messenger and other chat apps. In: *Bbc.com* [online]. web: BBC, 2018 [cit. 2019-12-17]. Dostupné z: https://www.bbc.com/news/newsbeat-43485511

[11] Tinder security. *Gotinder.com* [online]. gotinder.com: Tinder, 2019 [cit. 2019-12-17]. Dostupné z: https://www.gotinder.com/security?locale=cs

[12] Threema crypto differences. *Threema.ch* [online]. threema.ch: threema, 2019 [cit. 2019-12-17]. Dostupné z: https://threema.ch/en/faq/crypto_differences

[13] Ricochet - Most Secure Peer-to-Peer Encrypted Messenger that Sends No Metadata. *Thehackernews.com* [online]. thehackernews.com: thehackernews, 2016 [cit. 2019-12-17]. Dostupné z: https://thehackernews.com/2016/02/ricochet-secure-messenger.html

[14] Telegram - MTProto Mobile Protocol. *Core.telegram.org* [online]. core.telegram.org: core.telegram.org, 2019 [cit. 2019-12-17]. Dostupné z: https://core.telegram.org/mtproto

[15] Demystifying the Signal Protocol for End-to-End Encryption (E2EE). *Cloudboost.io* [online]. blog.cloudboost.io: cloudboost.io, 2017 [cit. 2019-12-17]. Dostupné z: https://blog.cloudboost.io/demystifying-the-signal-protocol-for-end-to-end-encryption-e2ee-3e31830c456f

[16] , Tyagi, Nirvan and Gilad, Yossi and Leung, Derek and Zaharia, Matei and Zeldovich a Nickolai. Proceedings of the 26th Symposium on Operating Systems Principles. , Tyagi, Nirvan and Gilad, Yossi and Leung, Derek and Zaharia, Matei and Zeldovich a Nickolai. *Proceedings of the 26th Symposium on Operating Systems Principles: Stadium: A Distributed Metadata-Private Messaging System*. SOSP '17. Shanghai, China: ACM, 2017, s. 440. ISBN 978-1-4503-5085-3. Dostupné také z: https://dl.acm.org/citation.cfm?id=3132783

[17] Most Popular Instant Messengers 1997 - 2019. In: *https://sec.gov* [online]. sec.gov: sec.gov, 2019, 21.9.2019 [cit. 2019-12-17]. Dostupné z: https://www.youtube.com/watch?v=KbWfzyQBWrU

[18] DANEZIS, George, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Rodica Tirtea a Stefan Schiffner. Privacy and Data Protection by Design - from policy to engineering. *CoRR*. 2015, **2015**, 79. Dostupné také z: http://arxiv.org/abs/1501.03726

[19] GRPC docs. *Https://grpc.io* [online]. grpc.io: grpc.io, 2019 [cit. 2019-12-17]. Dostupné z: https://grpc.io/docs/

[20] GraphQL spec. *Graphql.github.io* [online]. graphQL: graphQL, 2019 [cit. 2019-12-17]. Dostupné z: https://graphql.github.io/graphql-spec/

[21] TOR docs. *Torproject.org* [online]. torproject: torproject, 2019 [cit. 2019-12-17]. Dostupné z: https://2019.www.torproject.org/docs/documentation.html.en

[22] Android cryptography lib. *Developer.android.com* [online]. developer.android.com: developer.android.com, 2019 [cit. 2019-12-17]. Dostupné z: https://developer.android.com/guide/topics/security/cryptography

[23] WolfSSL lib. *Wolfssl.com* [online]. wolfssl.com: wolfssl, 2019 [cit. 2019-12-17]. Dostupné z: https://www.wolfssl.com/docs/

[24] Spongycastle lib. *Github* [online]. github.com: github, 2019 [cit. 2019-12-17]. Dostupné z: https://rtyley.github.io/spongycastle/

[25] Angelo De Caro a Vincenzo Iovino. , Angelo De Caro a Vincenzo Iovino. *IEEE Symposium on Computers and Communications (ISCC)*. Kerkyra, Greece: ieee-explore, 2011, s. 122. ISBN 978-1-4577-0678-3.

[26] Cryptography whitepaper. *Threema* [online]. 2019, 2019, 22 [cit. 2020-03-08]. Dostupné z: https://threema.ch/press-files/cryptography_whitepaper.pdf

[27] Messener secret conversations technical whitepaper. *Fbnewsroomus* [online]. 2017, 15 [cit. 2020-03-08]. Dostupné z: https://fbnewsroomus.files.wordpress.com/2016/07/messenger-secret-conversations-technical-whitepaper.pdf

Zoznam symbolov, veličín a skratiek

GPS	Global Positioning System
VK	Verejný kľúč
SK	Súkromný kľúč
RSA	Rivest–Shamir–Adleman. Asymetrická šifra, používa rozdielny kľúč na šifrovanie (VK) a dešifrovanie (SK) je prvým algoritmom, ktorý je vhodný pre podpisovanie aj pre šifrovanie. Je založené na predpoklade, že nejestvuje algoritmus, ktorý si poradí s faktorizáciou veľkých čísel.
DH	Diffie-Hellman protokol slúžiaci na dohodnutie kľúča cez verejný kanál
AES	Advanced Encryption Standard - Štandardizovaný algoritmus pre Symetrickú šifru
E2E	End to End
JWT	JSON Web Token
TLS	Transport Layer Security
SHA	Secure Hash Algorithm
TOR	The Onion Router
VPN	Virtual Private Network
API	Application programming interface
IM	Instant Messaging
DB	Databáza

Zoznam príloh

priložené subory obsahujú:

- zdrojové súbory k projektu aplikácii
- inštalačný súbor aplikácie, ktorý aplikáciu pripojí k môjmu google serveru, kde bude všetko nakonfigurované.
- zdrojové súbory užívateľského serveru
- zdrojové súbory serveru miestností
- zdrojové súbory serveru kľúčov
- zdrojové súbory autorizačného serveru
- zdrojové súbory serveru notifikácií

Na dáta uložené v DB na google serveri je možné použiť http://34.66.74.230:5050 a prihásiť sa údajmi:
meno="youremail@yourdomain.com"
heso="vutbrcz"

Inštalačku aplikácie je nutné nainštalovať na Android smartfón

Príprava serveru je nasledovná požiadavky OS Linux s nainštalovanou podporou Docker a v nej doinštalovať jazyk Golang vytvorené prístupy k lokálnej PostgreSQL, s ktorou budú servre komunikovať, každý server obsahuje súbor s cestou */DB/constants/Constants.go* kde je nutné vyplniť prístupy do databázy.

V priečinku **Grobardo_server_device_tokens** spustiť nasledujúcie príkazy
docker build . -t grpc_device_tokens_8083:1.0 -f dockerfile
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8083:8083 [ID of image]

docker build . -t grpc_device_tokens_8883:1.0 -f fundockerfile
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8883:8883 [ID of image]

V priečinku **Grobardo_server_rooms** spustiť nasledujúcie príkazy
docker build . -t grpc_rooms_8082:1.0 -f dockerfile
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8082:8082 [ID of image]

docker build . -t grpc_rooms_func_8882:1.0 -f fundockerfile

a výsledný image spustiť príkazom
sudo docker run -it -d -p 8882:8882 [ID of image]

V priečinku **Grobardo_server_jwt** spustiť nasledujúcie príkazy
docker build -t grpc_auth_8084:1.0 .
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8084:8084 [ID of image]

V priečinku **Grobardo_server_messages** spustiť nasledujúcie príkazy
docker build . -t grpc_messages_8080:1.1 -f dockerfile
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8080:8080 [ID of image]

V priečinku **Grobardo_server_notification** spustiť nasledujuce príkazy
jedná sa o nepovinný server, ak nebude v systéme, nebudu na zaraidenia chodiť notifikácie.
docker build -t python_notifications_8085:1.0 .
a výsledný image spustiť príkazom
sudo docker run -it -d -p 8085:8085 [ID of image]